NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

3D VISUALIZATION OF AN INVARIANT DISPLAY STRATEGY FOR HYPERSPECTRAL IMAGERY

by

Kang Suk Kim

December 2002

Co-Advisors: Richard C. Olsen
Donald P. Brutzman

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE 3D Visualization of an Invariant Display Strategy for Hyperspectral Imagery		**5. FUNDING NUMBERS**
6. AUTHOR Kang Suk Kim		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATIO N REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the U.S. Department of Defense or the U.S. Government.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**

13. ABSTRACT

Spectral Imagery provides multi-dimensional data, which are difficult to display in standard three-color image formats. Tyo, et al. (2001) propose an invariant display strategy to address this problem. This approach is to mimic the dynamics of human perception. The dimensionality of the data are reduced by using a Principal Component (PC) transformation, and then displayed by making used of a Hue, Saturation, and Value (HSV) display transform.

This study addresses the PC transformation strategy, looks for a global eigenvector via 3D visualization of HSV color space information, and examines the suggested algorithm to provide the most intuitive display. The user interface created in this thesis is capable of computing the necessary implementation of the proposed strategy, viewing selected Region of Interest (ROI) in HSV color space model in 3D, and viewing the 2D resultant image. A demonstration application uses Java language including Java2D, Xj3D Player, Document Object Model (DOM) Application Program Interfaces (API), and Extensible 3D Language (X3D). The Java2D API enables the user to load imagery, process data, and render results in a two-dimensional (2D) view. Xj3D and DOM APIs are introduced to visualize Tyo's invariant display strategy in three-dimensional (3D) views and then to save results as X3D scenes. These techniques appear to be inherently valuable and can serve as the basis for further research.

Through this thesis, 3D visualization of the proposed algorithm successfully showed PC transformed data does form a conical shape in HSV color space. Also, a comparison of PC transformed data with HSV color space revealed the hue angle needed to be adjusted. The application of this adjustment to multiple scenes produced consistent results. However, this hue adjustment left other scene elements in non-ergonomic colors and brought up the issue of further enhancement of the algorithm.

14. SUBJECT TERMS Hyperspectral Imagery, 3D Visualization, X3D, Java Xj3d API			**15. NUMBER OF PAGES** 106
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 **Standard Form 298 (Rev. 2-89)**
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

3D VISUALIZATION OF AN INVARIANT DISPLAY STRATEGY FOR HYPERSPECTRAL IMAGERY

Kang Suk Kim
Captain, Korean Army
B.S., Korean Military Academy, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED PHYSICS AND COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2002

Author:	Kang Suk Kim
Approved by:	Richard C. Olsen Co-Advisor
	Donald P. Brutzman Co-Advisor
	Peter J. Denning Chair, Department of Computer Science
	William B. Maier II Chair, Department of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Spectral Imagery provides multi-dimensional data, which are difficult to display in standard three-color image formats. Tyo, et al. (2001) propose an invariant display strategy to address this problem. This approach is to mimic the dynamics of human perception. The dimensionality of the data are reduced by using a Principal Component (PC) transformation, and then displayed by making used of a Hue, Saturation, and Value (HSV) display transform.

This study addresses the PC transformation strategy, looks for a global eigenvector via 3D visualization of HSV color space information, and examines the suggested algorithm to provide the most intuitive display. The user interface created in this thesis is capable of computing the necessary implementation of the proposed strategy, viewing selected Region of Interest (ROI) in HSV color space model in 3D, and viewing the 2D resultant image. A demonstration application uses Java language including Java2D, Xj3D Player, Document Object Model (DOM) Application Program Interfaces (API), and Extensible 3D Language (X3D). The Java2D API enables the user to load imagery, process data, and render results in a two-dimensional (2D) view. Xj3D and DOM APIs are introduced to visualize Tyo's invariant display strategy in three-dimensional (3D) views and then to save results as X3D scenes. These techniques appear to be inherently valuable and can serve as the basis for further research.

Through this thesis, 3D visualization of the proposed algorithm successfully showed PC transformed data does form a conical shape in HSV color space. Also, a comparison of PC transformed data with HSV color space revealed the hue angle needed to be adjusted. The application of this adjustment to multiple scenes produced consistent results. However, this hue adjustment left other scene elements in non-ergonomic colors and brought up the issue of further enhancement of the algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

ACKNOWLEDGEMENTS

At some point, you start writing acknowledgements and taking them for granted. Then, you realize that this is the only section that most of your family will read and understand, and you slow down and get them right.

First, I would like to thank to my country, Republic of Korea, which allowed me to have such a wonderful opportunity to study leading-edge technology both in Computer Science and Applied Physics.

For the technical folks, Chris Olsen spent many late afternoons with me to provide such details about this thesis topic with enthusiastic academic excitement. Don Brutzman guided me to this necessary level of knowledge, way before this research reached the culmination point.

My family is always amusing and always interested. My wife, Sunju Lee is always there to state the obvious in a way that makes me laugh. Not only was she interested in my work, but she also understood the complete content of this study much quicker than I did. In addition to her understanding, she contributed to this work by making some of figures presented here more illustrative, using her own architecture design technique. She also inspired me through pleasant discussion such that I could organize the overall structure.

She has lived with a husband who has been studying for more hours a day than he spends with her, for nearly two years, and has always loved and supported me. That is saying a lot, because I am a royal pain most of the time. I love you, honey.

Last and most important, to the Lord who got me this far: even so, come, Lord Jesus. I am ready to go home.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

Hyperspectral Image (HSI) data-collection technology emerged in 1987 in combination with the development of sensor technology. The defining characteristic of HSI is a tremendous increase in the number of frequency bands in which the instrument can collect data. Upon the collection of such data, difficulties arose when analysts tried to map these available bands to Red, Green, and Blue (RGB) to view the image data and to extract the available information. To overcome this difficulty, a number of varied mapping strategies were introduced. One analysis strategy was developed by Tyo, et al. (2001). It uses Principal Component Analysis (PCA) to rotate the data into a coordinate space, which can be used to display the data.

This thesis demonstrates how to visualize the conical shape of data mapped via PC to verify the HSI mapping strategy, and also identifies a global statistics by applying this strategy to other image data.

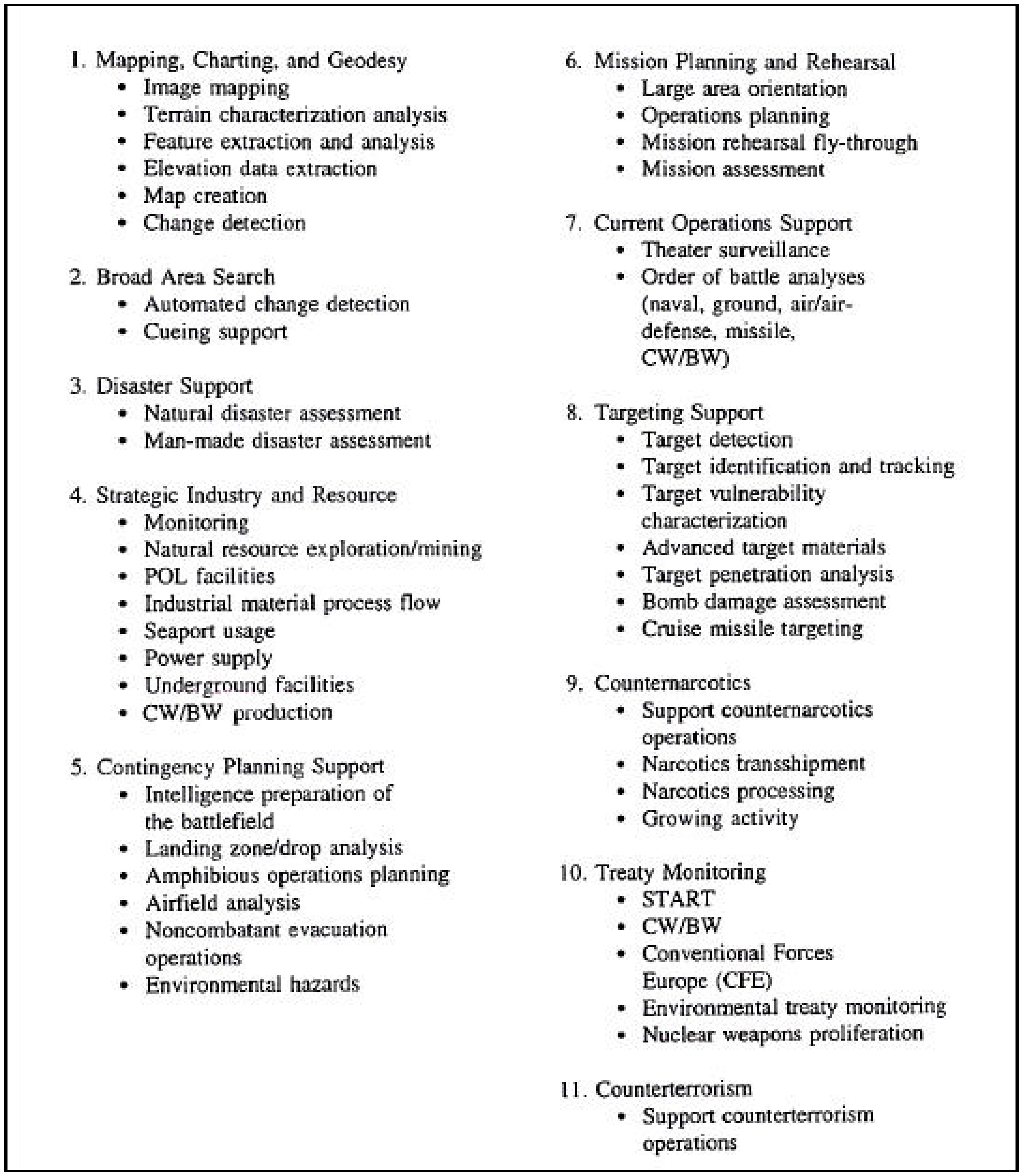
B. MOTIVATION

1. Merit of Hyperspectral Image (HSI) in Defense Application

Hyperspectral data have potential applicability for many different defense-related problems and tasks. For any given task, the critical factors that determine the usefulness of the spectral data include the phenomena that can be observed, the sensor parameters that determine how well the scene characteristics can be sampled and defined and, finally, the amount of relevant information that can be extracted from the remotely sensed data.

To help establish requirements for new systems, or to determine the applicability of existing systems, studies have identified the relevant factors and information elements for an Advanced Land Remote Sensing System (ALRSS), as a possible follow-on to the Landsat series of multispectral satellites (Anderson et al., 1994). Such studies identified defense applications and organized them into the eleven major categories shown in Table 1.1. Since information needs vary and depend on particular circumstances, it is difficult to make a single prioritized list of applications out of those presented in this thesis. (Anderson et al., 1994)

Table 1.1. Defense Application of Hyperspectral Sensors. (Anderson. et al., 1994)



1. Mapping, Charting, and Geodesy
    - Image mapping
    - Terrain characterization analysis
    - Feature extraction and analysis
    - Elevation data extraction
    - Map creation
    - Change detection
2. Broad Area Search
    - Automated change detection
    - Cueing support
3. Disaster Support
    - Natural disaster assessment
    - Man-made disaster assessment
4. Strategic Industry and Resource
    - Monitoring
    - Natural resource exploration/mining
    - POL facilities
    - Industrial material process flow
    - Seaport usage
    - Power supply
    - Underground facilities
    - CW/BW production
5. Contingency Planning Support
    - Intelligence preparation of the battlefield
    - Landing zone/drop analysis
    - Amphibious operations planning
    - Airfield analysis
    - Noncombatant evacuation operations
    - Environmental hazards
6. Mission Planning and Rehearsal
    - Large area orientation
    - Operations planning
    - Mission rehearsal fly-through
    - Mission assessment
7. Current Operations Support
    - Theater surveillance
    - Order of battle analyses (naval, ground, air/air-defense, missile, CW/BW)
8. Targeting Support
    - Target detection
    - Target identification and tracking
    - Target vulnerability characterization
    - Advanced target materials
    - Target penetration analysis
    - Bomb damage assessment
    - Cruise missile targeting
9. Counternarcotics
    - Support counternarcotics operations
    - Narcotics transshipment
    - Narcotics processing
    - Growing activity
10. Treaty Monitoring
    - START
    - CW/BW
    - Conventional Forces Europe (CFE)
    - Environmental treaty monitoring
    - Nuclear weapons proliferation
11. Counterterrorism
    - Support counterterrorism operations

2. Previous Work and Problem Statement

Even though HSI has many useful applications, difficulties occur when analysts try to map diverse data into RGB color space due to the increased available spectral bands in HSI (Dierson, 2000). The most common way to map HSI data into RGB color space is to represent the Principal Components data as RGB components. The outcome of this image-rendering process is often rendered in false color. This concept goes back to early Landsat studies, and the development of the standard Tasseled Cap Transform (Richard, 1993). Unfortunately, the appearance of imagery causes difficulty when observers try to identify objects in the imagery, as illustrated in Figure 1.1.

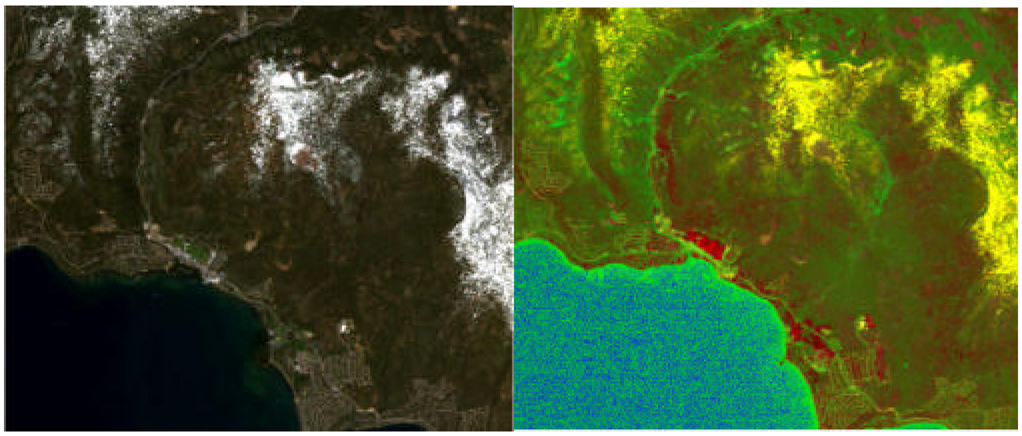


Figure 1.1. The Left is the true color image of Lake Tahoe acquired from Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). The selected wavelengths are 647.65nm for red, 549.2nm for green, and 431.71nm for blue RGB components. In contrast, the right side of the image appears in false color. The selected bands for RGB components are PC1, PC2, and PC3 respectively.

As seen, the PC data is mapped into RGB color bands on the right side of Figure 1.1. This image shows the snow area in yellow, and the blue downtown area in red. Other objects, such as a lake, a road, and a golf course do not appear in intuitive natural colors, either. This false-color appearance makes unsophisticated analysis difficult without *a priori* knowledge of HSI artifacts.

To overcome this difficulty, a new mapping strategy was introduced (Tyo, et al., 2001). They recognized the fact that the first three eigenvectors of remotely sensed signal data sets are the most important in describing any scene. By using this fact, a Principal Component (PC)-based mapping strategy was introduced. According to the PC color-mapping strategy, a combination of the first three principal component bands will appropriately depict most of the information in a scene. It provides an easy way to perform a first-order supervised classification of hyperspectral imagery. It was further shown that the first eigenvector would generally be related to the mean solar radiance, but the second, third and subsequent eigenvectors depend on the specific contents of the image. This thesis implements the approach defined by Tyo in a manner, which provides a consistent and intuitive display.

3. What is the Visualization Problem?

The main visualization problem occurs at the time when this strategy transforms calculated Hue, Saturation, and Value (HSV) data to RGB data. HSV color space is created from brightness value, the Red/Green (R/G) channel and the Blue/Yellow (B/Y) channel. Amazingly, it has been found that the three Principal Components (PCs) match the brightness value, R/G value, and B/Y value, respectively (Krauskopf, 1982). A human brain senses three channels, creates HSV color space, and transforms it to RGB internally to perceive. This algorithm calculates three PC datasets from the scene statistics, produces HSV, and transforms into RGB according to conversion. The calculation of PCs has been rigorously examined in previous work, and conversion from HSV to RGB is a well-established algorithm. The main visualization problem is that the HSV product has not been applied to the actual RGB image. Thus, this thesis will apply the HSV product based upon the proposed strategy, and verify what possible additional arrangements are required.

C. OBJECTIVES

This thesis addresses the following topics:

- What are the overall procedures for Remote Sensing? How does the imagery data flow from collection point to the final analysis point? What kinds of image processing are available?
- What are hyperspectral image and the image data formats used in Remote Sensing?
- What is the display strategy proposed by Dr. Scott Tyo? What is the Principal Component Transform (PCT)? What role does PCT play in the suggested strategy? What is the Eigenvector and what is it for?
- What is the relationship between the HSV color model and the principal component transformation?
- How can these techniques be rendered and analyzed in 3D for better comprehension and analysis?

- How can XML, X3D and Java tools be applied for visualizing this problem?

D. ORGANIZATION OF THESIS

Chapter II describes related work and the physics of Remote Sensing (RS). Chapter III covers the related graphics and visualized issues. Chapter IV provides the detailed problem statement in this thesis. Chapter V examines how the scene is generated and rendered. Chapter VI explains how the visualized Principal Component (PC) data is applied to verify and improve Tyo's algorithm. Furthermore, it provides what it means and applies the result to the other image data. Chapter VII summarizes conclusions and presents recommendations for possible future work.

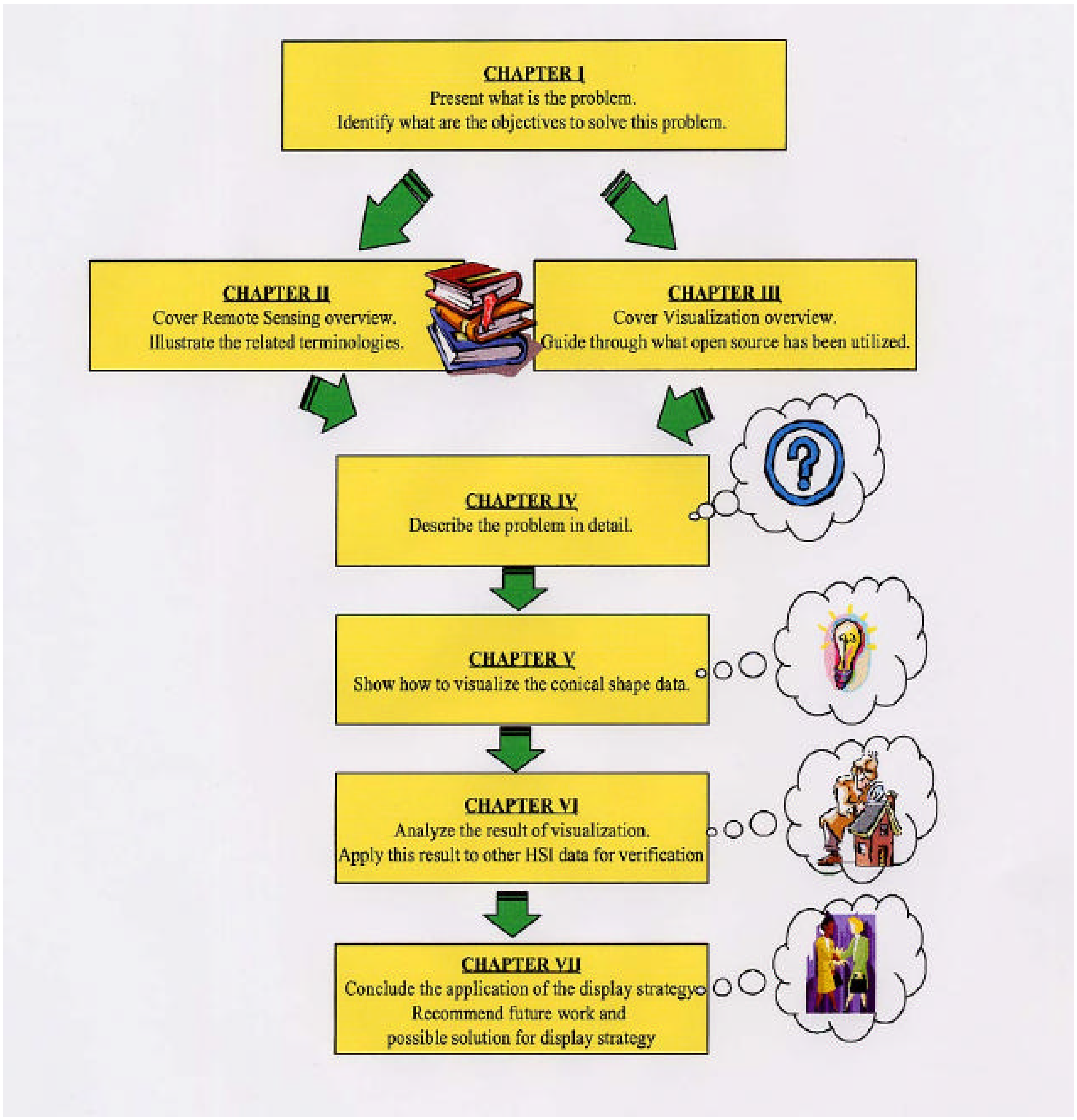

CHAPTER I
Present what is the problem.
Identify what are the objectives to solve this problem.
CHAPTER II
Cover Remote Sensing overview.
Illustrate the related terminologies.
CHAPTER III
Cover Visualization overview.
Guide through what open source has been utilized.
CHAPTER IV
Describe the problem in detail.
CHAPTER V
Show how to visualize the conical shape data.
CHAPTER VI
Analyze the result of visualization.
Apply this result to other HSI data for verification
CHAPTER VII
Conclude the application of the display strategy
Recommend future work and
possible solution for display strategy

Figure 1.2. Organization of Thesis.

II. BACKGROUND AND RELATED WORK: REMOTE SENSING

A. OVERVIEW

What exactly is **remote sensing**? For the purpose of this study, the following definition is used:

> Remote sensing is the science of acquiring information about the Earth's surface without actually being in contact with it. This is done by sensing and recording reflected or emitted energy and processing, analyzing, and applying that information. (Natural Resources Canada)

In much of remote sensing, the process involves an interaction between incident radiation and the targets of interest. This is exemplified by the use of imaging systems where the following seven elements are involved. Note, that remote sensing also involves the sensing of emitted energy and the use of non-imaging sensors. Figure 2.1 illustrates the major steps in the remote sensing process: (A) energy source, (B) radiation, (C) target illumination, (D) sensor/platform, (E) data transmission, (F) data analysis, and (G) applications for image interpretation and analysis.

Referring to Figure 2.1, the energy source or illumination (A) is the first requirement for remote sensing to have an energy source, which illuminates or provides electromagnetic energy to the target of interest. Radiation interacts with the atmosphere (B) as the energy travels from its source to the target, and it will come into contact with and interact with the atmosphere through which it passes. This interaction also takes place a second time as the energy travels from the target to the sensor. Interaction with the target (C), once the energy makes its way to the target through the atmosphere, interacts with the target depending on the properties of both the target and the radiation. The recording of energy by the sensor (D) occurs after the energy has been scattered by, or emitted from the target, and a sensor (remote - not in contact with the target) collects and records the electromagnetic radiation. The transmission, reception, and processing (E) occurs when the energy recorded by the sensor has to be transmitted, often in electronic form, to a receiving and processing station where the data are processed into an image (hardcopy and/or digital data). Interpretation and Analysis (F) occurs when the processed image is interpreted, visually and/or digitally or electronically, to extract information about the target, which was illuminated. Application (G), the final element of the remote sensing process, is achieved when the information is applied, the user has been able to extract it from the imagery about the target in order to better understand it, reveals some new information, or assists in solving a particular problem. These seven elements comprise the remote sensing process from beginning to end.

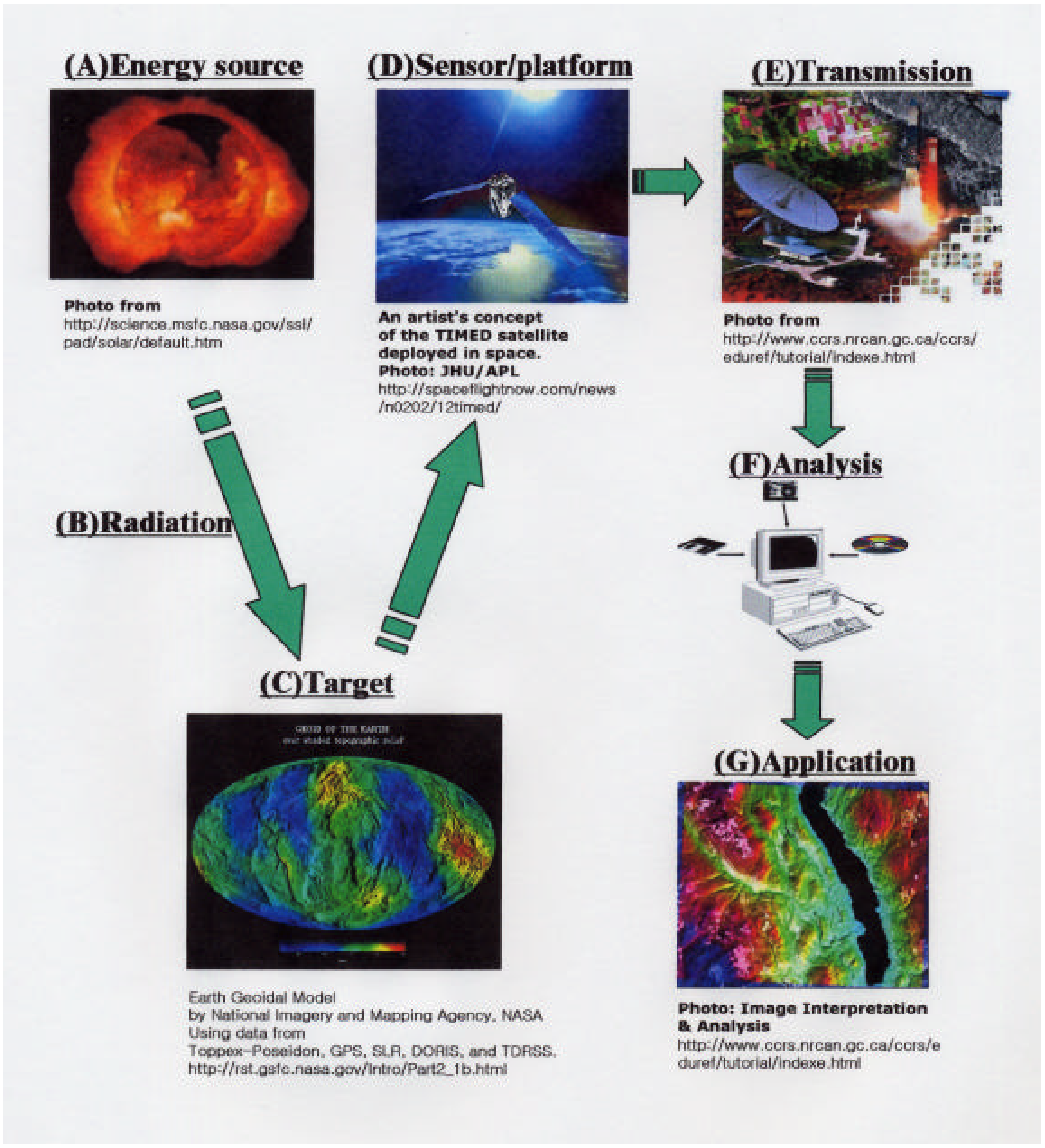

(A)Energy source
Photo from
http://science.msfc.nasa.gov/ssl/pad/solar/default.htm
(D)Sensor/platform
An artist's concept of the TIMED satellite deployed in space.
Photo: JHU/APL
http://spaceflightnow.com/news/n0202/12timed/
(E)Transmission
Photo from
http://www.ccrs.nrcan.gc.ca/ccrs/eduref/tutorial/indexe.html
(B)Radiation
(F)Analysis
(C)Target
Earth Geoidal Model
by National Imagery and Mapping Agency, NASA
Using data from
Toppex-Poseidon, GPS, SLR, DORIS, and TDRSS.
http://rst.gsfc.nasa.gov/Intro/Part2_1b.html
(G)Application
Photo: Image Interpretation & Analysis
http://www.ccrs.nrcan.gc.ca/ccrs/eduref/tutorial/indexe.html

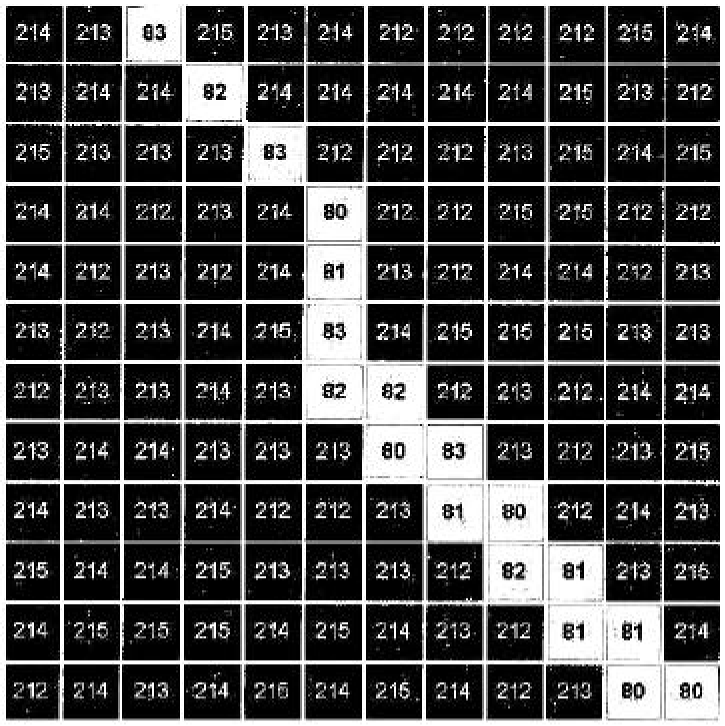
Figure 2.1.	Remote Sensing Process.

B. IMAGE ACQUISITION

Many different systems have been developed over the last fifty years, each of which are designed to acquire an image directly in digital form or to acquire an analog image and later digitize it. In either case, the final product is the same. A typical system is based upon a moving sensor or lens that aims the instrument's field of view over a portion of the earth's surface (Sanchez, 1999). The sensors on the instrument generate an electrical signal that varies in intensity according to the brightness of the object in view. These sensors are often designed to image a separate region of the electromagnetic spectrum. This is frequently accomplished through filters. Each separate electrical current is then scaled into discrete units with a predefined range. A numeric range of 0 to 8 is often used to represent pixels in 256 shades of gray. For example, the Thematic Mapper (TM) instrument uses a range of 0 to 255.

C. DIGITAL NUMBER (DN)

The digital form of image data has exceptional advantages versus its analogue form of image data. These advantages range from the perspective of data transmission to image acquisition, to the ground data recipient and to data manipulation for analysis. The sensor performs the conversion from analogue to digital data. In this digitizing process, the output from an electronic sensor is converted into a set of numerical values. These numerical values are known as Digital Numbers (DN) and represent a current radiance intensity level in a sensor such as Charge Coupled Device (CCD) (Sanchez, 1999). Figure 2.2 shows example values that can be represented using different numbers of binary digits. In this thesis, the 16 bit signed integers used range from –32768 to 32767.



214	213	83	215	213	214	212	212	212	212	215	214
213	214	214	82	214	214	214	214	214	215	213	212
215	213	213	213	83	212	212	212	213	215	214	215
214	214	212	213	214	80	212	212	215	215	212	212
214	212	213	212	214	81	213	212	214	214	212	213
213	212	213	214	215	83	214	215	215	215	213	213
212	213	213	214	213	82	82	212	213	212	214	214
213	214	214	213	213	213	80	83	213	212	213	215
214	213	213	214	212	212	213	81	80	212	214	213
215	214	214	215	213	213	213	212	82	81	213	215
214	215	215	215	214	215	214	213	212	81	81	214
212	214	213	214	215	214	215	214	212	213	80	80

Figure 2.2. The Example of Digital Number (DN) from Multispectral Imagery Reference Guide. (From: Belokon, 1997)

D. HYPERSPECTRAL IMAGERY (HSI)

The two general classes of remote imaging sensors are active (transmit-receive) and passive (receive only) (ENVI User Reference). Hyperspectral Imagery (HSI) sensors are passive and collect/record data in hundreds of spectral bands covering the spectrum from ultraviolet to infrared. The increased number of sensor bands in HSI provides higher spectral resolution and more opportunities to detect subtle spectral differences in signatures that are too narrow to be differentiated on multispectral imagery. The significant increase in data to be analyzed has resulted in the development of completely different methods of analysis from that traditionally used for multispectral imagery. Hyperspectral Imaging (HSI) remains an emerging, developing technology and is currently being tested with the NASA Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) and the DoD Hyperspectral Digital Imagery Collection Experiment (HYDICE) programs, as well as several commercial aircraft sensors. (Multispectral User Guide, 1995). Depending on which sensors were used when the data was collected, the number of bands of data in this study varies from 210 to 300.

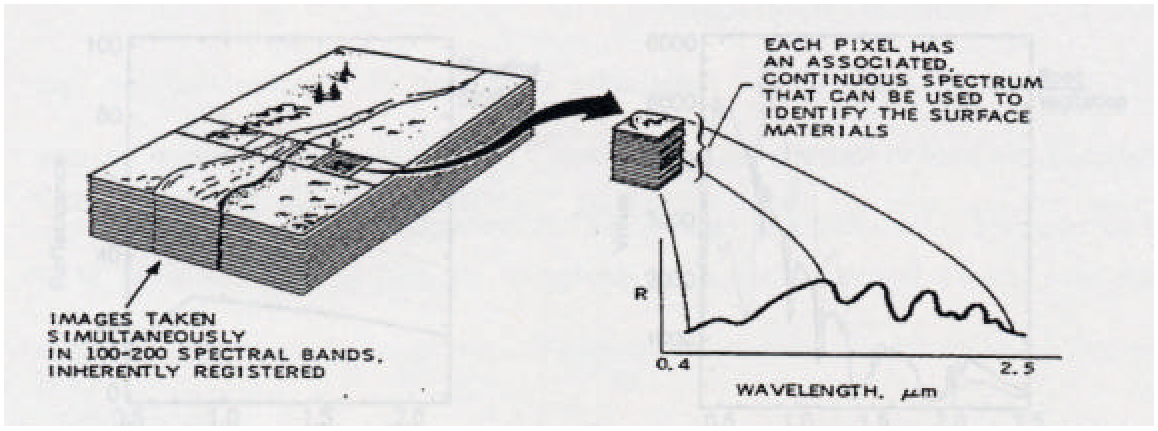

EACH PIXEL HAS AN ASSOCIATED, CONTINUOUS SPECTRUM THAT CAN BE USED TO IDENTIFY THE SURFACE MATERIALS
IMAGES TAKEN SIMULTANEOUSLY IN 100-200 SPECTRAL BANDS, INHERENTLY REGISTERED
R
0.4
2.5
WAVELENGTH, μm

Figure 2.3. Hyperspectral Image (HSI) Data Processing. (From: Vane and Goetz, 1988)

E. DATA FORMATS

Digital image representations can be achieved using vector or raster formats. In vector graphics, visual objects are defined in terms of lines, each of which is given a position, length and direction. Raster-graphics images, on the other hand, are a collection of dots, usually called pixels, organized in rows and columns. Remotely sensed images are almost invariably presented in raster format (Sabin, 1987). In fact, the notion of digital image data usually assumes that the image is a pattern of discrete and independent dots, each of which is represented by a DN. Image data organized by bands is usually structured into one of three image formats: Band Interleaved by Pixel (BIP), Band Interleaved by Line (BIL) or Band Sequential (BSQ). In this study, all the input data before manipulation is structured in BSQ for file input convenience, and data after processing are saved in BIP for file output convenience upon complete computation.

1. Band Interleaved by Pixel (BIP) Format

In BIP format, the banded data are stored in pixel-major order. Figure 2.3 graphically illustrates how a scene is stored in corresponding bands. Although there is not many image display situations in which the BIP format is advantageous, output of data in other work are in BIP format. This is because it is computationally advantageous to compute each pixel's color using three different bands immediately after loading the data into memory. This is a good format when the imagery data only consists of a small number of bands and all three bands need to be mapped to RGB bands, particularly for very large images. This format is the native format for the Airborne Visible Infrared Imaging System (AVIRIS) sensor.

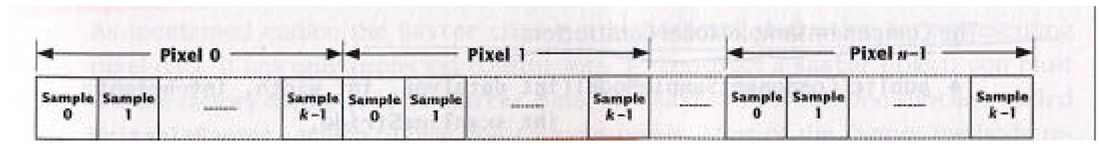


Figure 2.4 Band Interleaved by Pixel (BIP) Format. (From: Lyon, 1999)

2. Band Sequential (BSQ) Format

A frequently used format, and also the best one in terms of display ease, is the band sequential or BSQ. In this case, the banded data is stored in band-major order. That is, each image band appears consecutively in the data file with data elements ordered to correspond to each individual image pixel. Since the sensor radiation band is the natural unit of data organization, the BSQ format is also easy to implement. Figure 2.5 shows how a scene originally sensed in three different radiation ranges is stored in three corresponding bands, and how the band data is digitized and saved in BSQ format.

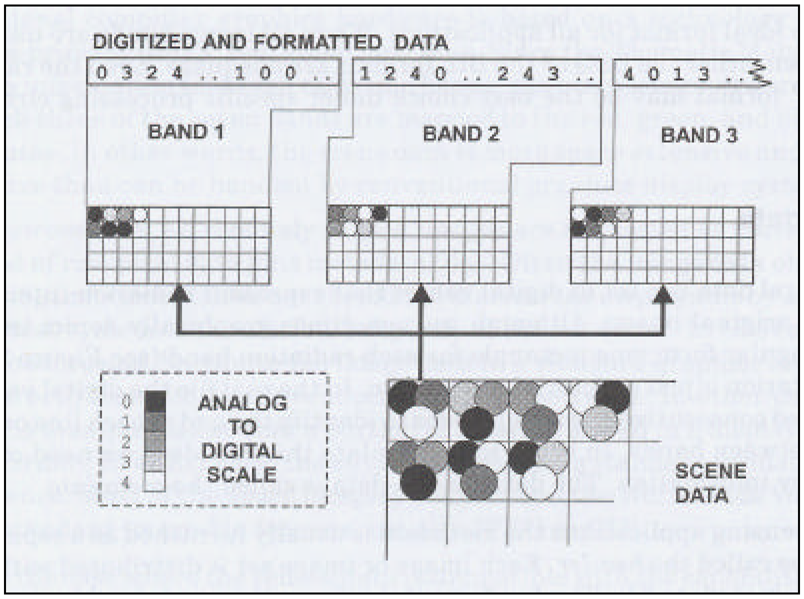


Figure 2.5. Band Sequential (BSQ) Format. (From: Sanchez, 1999)

Processing and image analysis routines are easier to code when the data is in BSQ format, especially in those cases in which certain bands are selected from a large data set. In the computer processing of this data, visualized algorithms often need to select three of these seven bands in order to map them to the three basic screen color attributes. BIP offers computation advantages for spectral analysis.

3. Band Interleaved by Line (BIL) Format

In the Band Interleaved by Line (BIL) format, the image scan line constitutes the organizing base. When the data is stored in line-major order, the image lines are the lines appearing consecutively in the data. This format is frequently the "native" format for push broom imaging spectrometers, such as Hyperspectral Digital Image Collection Experiment (HYDICE). Figure 2.6 shows a scene originally sensed in three different radiation ranges and stored in three corresponding bands as well as how the band data is digitized and saved in BIL format (Sanchez, 1999).

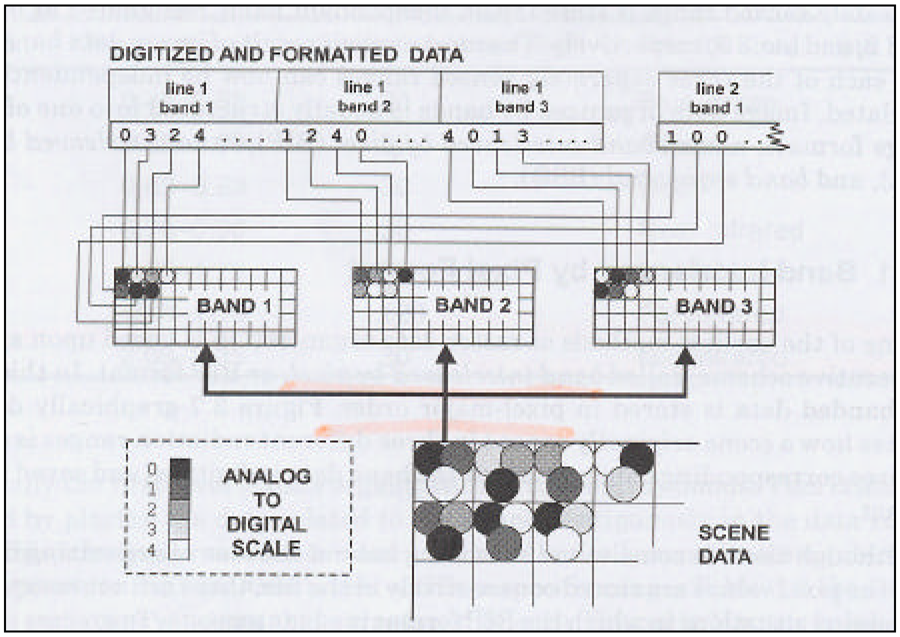


Figure 2.6 Band Interleaved by Line (BIL) Format.

F. NUMBER REPRESENTATION AND BYTE ORDER

There are groups of binary number representation: byte, integers and floating-point numbers. Every binary number is stored in a fixed amount of space, with fixed range of values and fixed precision (Fortner, 1996). The numbers are coded in a very effective way that is not 'human readable'.

The binary numbers except for "byte" representation take more than two bytes to represent the binary number. Also, number representation larger than a byte places bytes in one of two standardized orders: IEEE standard and Intel standard. Byte ordering is significant when the Digital Number (DN), introduced in Section C, uses integer or higher precision numbers. IEEE standard bytes are ordered from Most Significant bit First (MSF) to Least Significant bit First (LSF). On the other hand, the "Intel" standard is bytes that are ordered from LSF to MSF. The imagery can be received in either standard. This thesis uses the IEEE standard, when necessary that input data are corrected for byte order. The technique for this will be explained in Chapter V.

G. DIGITAL IMAGE PROCESSING

1. Overview of Information Extraction

Image restoration and enhancement utilize computational processes to provide corrected and improved images for study by human interpreters. The computer makes no inherent decisions in these procedures. However, processes that identify and extract information do utilize the computer's decision-making capability in order to identify and extract specific pieces of information. A human operator must instruct the computer on the parameter of interest, and must evaluate the significance of the extracted information as crucial steps in the visualization process.

Image-processing methods may be grouped into three functional categories. These are defined below, together with lists of typical processing routines (Sabin, 1999).

- Image restoration compensates for data errors, noise, and geometric distortions introduced during the scanning, recording, and playback operation: restoring periodic line dropouts, restoring periodic line striping, filtering of random noise, correcting for atmospheric scattering and correcting geometric distortions.
- Image enhancement alters the visual impact that the image has on the interpreter in a fashion that improves the information content: contrast enhancement, intensity, hue, saturation transformation, density slicing, edge enhancement, making digital mosaics and producing synthetic stereo images.

- Information extraction utilizes the decision-making capability of the computer to recognize and classify pixels on the basis of their digital signatures: producing principal-component (PC) images, producing ratio images, multispectral classification and producing change-detection images.

2. Principal Component Transform (PCT) Overview

The principal components transformation maps image data into a new, uncorrelated coordinated system or vector space (Swain, 1978). Independent of the class structure of the data, the Principal Components Transformation (PCT) process makes use of a global mean and global covariance. Moreover, in doing so, this data remapping produces a space in which the data has the most variance along its first axis, the next largest variance along a second mutually orthogonal axis, and so on. The later principal components are expected, in general, to show less variance. These principal components might thus be considered to contribute little to separability on occasion and may be safely ignored, thereby reducing the overall dimensionality of the classification space, and thus improving classification speed.

The following equations from 2.1 to 2.5 show the related mathematical definition of PC transformation. The PC transformation determines a linear transformation of a sample of points in N-dimensional space. The mean position of the pixels in N-dimensional space is defined by the expected value of the pixel x, according to Equation 2.1, where $\bar{x}$ is the mean pixel value and x_j is the individual pixel value of total number of N.

$$\text{Mean} = \bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j \tag{2.1}$$

While the mean vector is useful to define the average or expected position of the pixels in hyperspectral space, it is also of value to have a mean available by which their scatter or spread is described. This is the role of the covariance matrix, which is defined as Equation 2.2.

$$\text{covariance} = \frac{1}{N-1}\left[\sum_{j=1}^{N} x_j y_j - \frac{1}{N}\sum_{j=1}^{N} x_j \sum_{j=1}^{N} y_j\right] = \frac{1}{N-1}\sum_{j=1}^{N}\left(x_j - \bar{x}\right)\left(y_j - \bar{y}\right) \tag{2.2}$$

The covariance matrix is one of the most important concepts in the hyperspectral data. If there is a correlation between the responses in a pair of spectral bands, the corresponding off-diagonal element in the covariance matrix will be large by comparison to the diagonal terms. This behavior is also described in terms of the correlation matrix R whose elements are related to those of the covariance matrix by

$$\text{Correlation Coefficient} = r = \frac{N\sum_{j=1}^{N} x_j y_j - \sum_{j=1}^{N} x_j \sum_{j=1}^{N} y_j}{\left[N\sum_{j=1}^{N} x_j^2 - \left(\sum_{j=1}^{N} x_j\right)^2\right]^{1/2}\left[N\sum_{j=1}^{N} y_j^2 - \left(\sum_{j=1}^{N} y_j\right)^2\right]^{1/2}} \qquad (2.3)$$

It should be noted that the correlation coefficient and covariance are related by the standard deviation:

$$\text{correlation coefficient} = \text{covariance} \big/ s_x s_y \qquad (2.4)$$

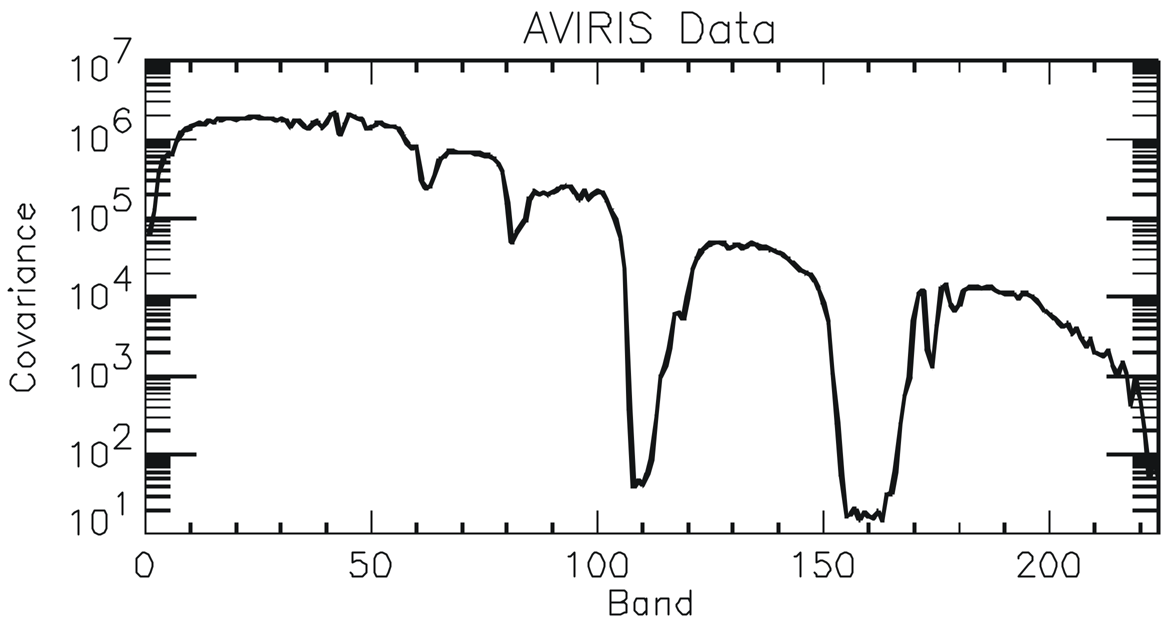
where the standard deviation and variation are given by

$$\text{Standard deviation in X and Y dimension} = s_x, s_y = \sqrt{\text{Variance}}, \text{ and} \qquad (2.5)$$

$$\text{Variance} = \frac{1}{N-1}\sum_{j=1}^{N}\left(x_j - \overline{x}\right)^2$$

3. Why Principal Component Transformation (PCT)?

Principal Components Transformation (PCT) is used to produce uncorrelated output bands, to segregate noise components, and to reduce the dimensionality of data sets. Since hyperspectral data bands are often highly correlated, the PCT is used to produce uncorrelated output bands. Essentially, this is done by finding a new set of orthogonal axes that have their origin at the data mean and are rotated so that the data variance is maximized.



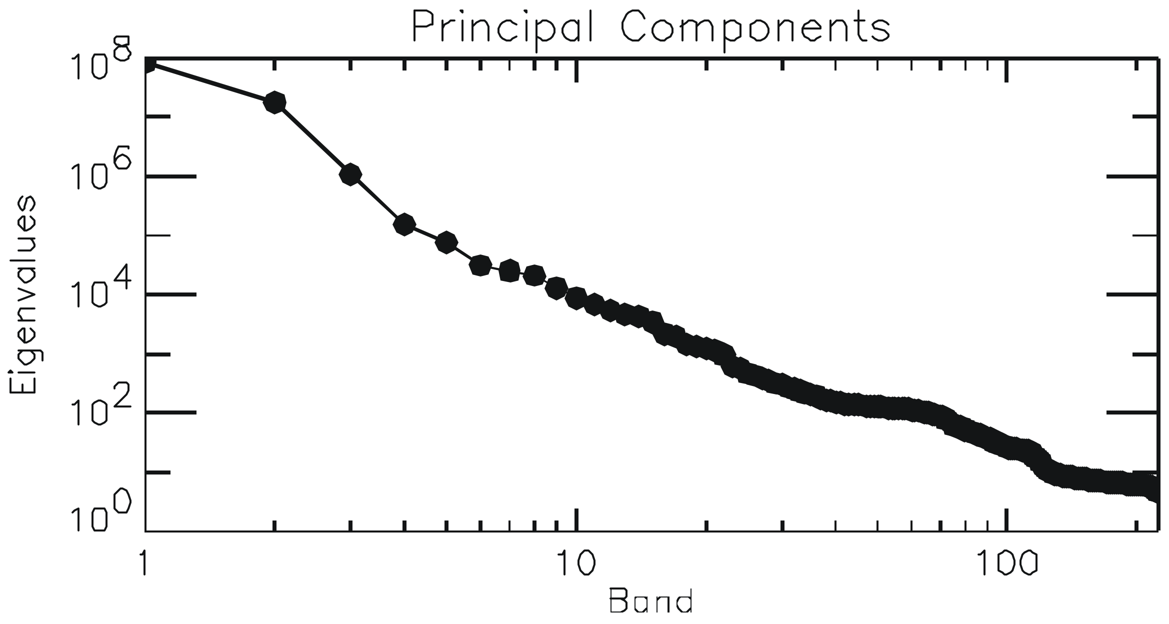


Figure 2.7. Comparison between Covariance and Eigenvector of AVIRIS data. Notice that the variance is scattered all over the band range in the original data and the variance information was concentrated into the first three Eigenvector after PC transformation.

The hyperspectral or vector character of most remote sensing image data renders it amenable to spectral transformations that generate new sets of image components or bands. These components then represent an alternative description of the data in which the new components of a pixel vector are related to its old brightness values in the original set of spectral bands via a linear operation. The transformed image may make evident features not discernable in the original data, or alternatively it might be possible to preserve the essential information content of the image, for a given application, with a reduced number of the transformed dimensions. The last point has significance for displaying the data in the three dimensions available on a color monitor or in color hardcopy, as well as for the transmission and storage of data (Richards, 1993).

PC bands are linear combinations of the original spectral bands and are typically uncorrelated. One can calculate the same number of output PC bands as input spectral bands. The first PC band contains the largest percentage of data variance, the second PC band contains the second largest data variance, and so on. The last PC bands appear noisy because they contain very little variance, mostly due to noise in the original spectral data.

4. Role of Eigenvectors in Image Data

Eigenvectors are the component-weighting coefficients (Richards, 1993). They provide the transform (rotation) into the new data space. The first component is tantamount to a total brightness image, whereas the alternate components highlight changes. It is the second, third and fourth components that are most striking in relation to the fine features of interest. These effects are easily verified by substituting typical spectral reflectance characteristics into the equations that generate the component. Each component is a linear combination of the original eight bands of data, where the respective weighting coefficients are the components of the corresponding eigenvector of the covariance matrix.

H. COLOR SPACE

HSV color space is often used for picking colors from a color wheel or palette, because HSV corresponds better to how people experience color than does the RGB color space (Jackson, 1997). As hue varies from 0 to 1, the corresponding colors vary from red, through yellow, green, cyan, blue, and magenta, back to red. As saturation varies from 0 to 1, the corresponding colors vary from unsaturated (shade of gray) to fully saturated (no white component). As value, or brightness, varies from 0 to 1, the corresponding colors become increasingly brighter. Figure 2.8 shows the HSV color space. This HSV representation is later used for visually comparing Principal Component (PC) eigenvector mappings in 3D space.

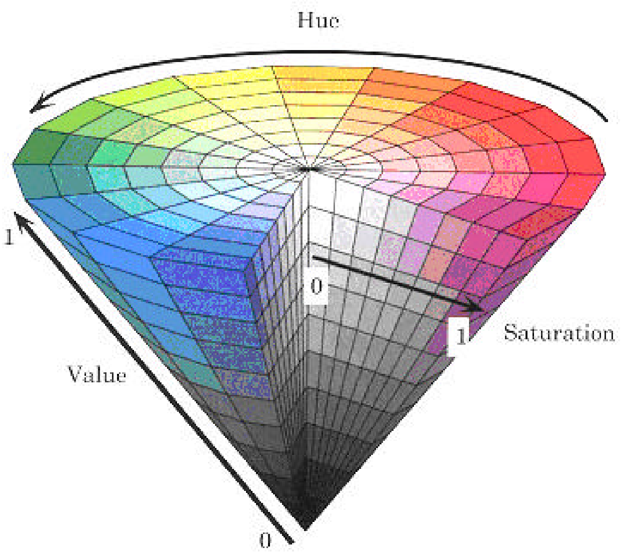


Figure 2.8. Hue, Saturation and Value (HSV) color space. (From: Image Processing Toolbox User's Guide, 1997)

THIS PAGE INTENTIONALLY LEFT BLANK

III. BACKGROUND AND RELATED WORK: VISUALIZATION

A. INTRODUCTION

This chapter presents related work for 2D and 3D visualization. The language used to build the application was Java and the reason for selecting it will be explained in the first section. The second section describes the necessary Java2D Application Program Interface (API), and next section describes the 3D related works including XML, X3D and Xj3D.

B. CHOICE OF LANGUAGE

The Java language is used because it provides several advantages over other languages in terms of platform independency, distributability, multithreadability and security (Liang, 2002). In addition to theses renowned capabilities, the built-in Java2D API provides excellent capability to access and manipulate an image pixel directly and easily (Lyon, 1999). The Java2D API uses the concept of a digital image representation concept for remote sensing, as introduced in the previous chapter. For 3D visualization, Java was chosen as well, since Java, in combination with VRML/X3D, is very powerful. (Brutzman, 1998)

The Java2D API is utilized to view the initial image data and processed image. X3D graphics are used to inspect the conical shape of the transformed data space. The next two sections discuss what these concepts are and demonstrate how they are implemented.

C. 2D VISUALIZATION

This section describes how Java2D is used to view binary image data. Each set of bytes for selected bands in this binary data represents the intensity of the red, green, and blue bands. The original binary data are all 16 bit short integers in the MS-DOS standard, which differs from the IEEE standard. All these binary data are in the BSQ format as mentioned in the previous chapter. This section addresses how initial binary data were converted to the IEEE standard, which is used in Java, and how they are packed into the appropriate format to be rendered for display.

1. Java2D API Overview

The Java2D API is a set of classes that can be used to create high-quality 2D graphics. It includes features such as geometric transformation, antialiasing, alpha compositing, image processing and bidirectional text layout, among many others. The Java2D is part of the core classes of the Java 2 platform. In other words, it is widely available with Java without requiring supplementary distributions (www.java.sun.com/java2d). The Java2D API introduces new classes in the following packages: java.awt and java.awt.image.

2. Displaying Images

a. Image Representation Overview

An image is a two-dimensional (2D) array of colors. Each element in the array is called a, "picture element" or a pixel (Knudsen, 1999). In general, there are two different approaches to generating image data. One approach is to treat the image as a drawing surface and use the methods of Graphics2D to render new objects superimposed onto the image. The other approach is to twiddle the bits of the image data. This can be useful in specific cases such as loading and saving images in single files, or mathematically analyzing image data to determine essential properties. The latter approach has been adopted for this thesis.

Images can be represented in several different ways: RGB data where red, green, and blue values for each pixel are stored as the elements of byte arrays; an RGB image where each pixel is represented by an integer that contains red, green, and blue values; or a 16-level grayscale image with 8 pixels stored in each element of an integer array. In this thesis, images of interest are represented by integer arrays that contain red, green and blue values.

The Java2D model for storing an image minimizes the storage required. (Rodrigues, 2001) A pixel is stored within a 32-bit integer (int) java primitive data type. The int consists of 4 packed bytes. These bytes represent the alpha, Red, Green and Blue (RGB) planes, as shown in Figure 31. The tightly packed storage technique available for 32-bit pixels is also shown in Figure 3.1.

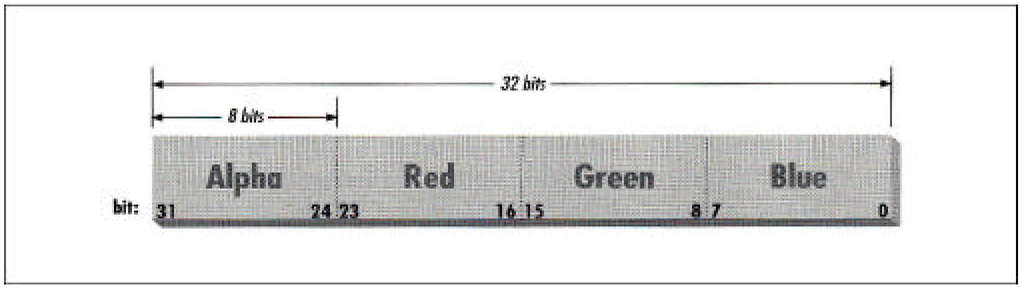


Figure 3.1. Pixel representation of image in Standard Red, Green and Blue (SRGB) format in Java. (From: Rodridgues, 2001)

b. java.awt.image.BufferedImage

Prior to Java2D, the use of an image producer and consumer model was the only method that allowed the manipulation of image data. The Java2D API, introduced in the Java 2 platform, includes an extension of Image, java.awt.image.BufferedImage, which allows direct access to the image's data. Since BufferedImage is a subclass of Image, a BufferedImage can work with any of Graphics2D's methods that accept an image. A BufferedImage provides improved control over the actual data comprising the image.

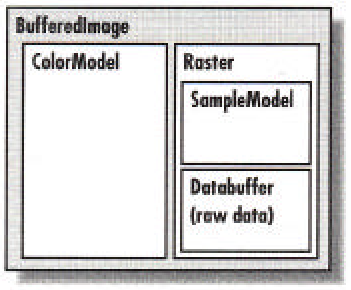


Figure 3.2. BufferedImage Class. (From: Knudsen, 1999).

3. Manipulating Image Data

A BufferedImage consists of two pieces: a Raster and a ColorModel. The Raster contains the actual image data. It is an array of pixel values. The ColorModel's job is to interpret the image data as colors. This ColorModel can translate the data values that come from the Raster into color objects. An RGB color model, for example, can interpret three data values as red, green and blue. The Raster itself is made up of two pieces: a DataBuffer and a SampleModel. The DataBuffer is a wrapper for the raw data arrays, which are byte, short or int arrays. The SampleModel knows how to extract the data values for a particular pixel from the DataBuffer. As utilizing assistance to the programmer, the Raster class has many static methods that create preconfigured Rasters, including their DataBuffers and SampleModels. Figure 3.3 shows code snippets where the data buffer is directly applied to the BufferedImage class.

Figure 3.3 illustrates the style of coding used in this thesis. The integer array named `int[] data` was used to represent each pixel in RGB color. Each pixel, `data[i]`, consists of 32 bits. Each integer contains the RGB color information. The first 8 bits from the right-most bit contain the blue color intensity information, which can be between 0 and 255. The next 8 bits from the end of the blue color information represent the green color information. Similarly, the red values were placed at the next 8 bits after the green information.

The BufferedImage class is somewhat complex as there are many different ways to represent the colors of pixels. However, using both a predefined color space type in the BufferedImage constructor, and the `setRGB()` method inherited from the java.awt.Image class, reduces the amount of coding relating to the BufferedImage class. This is a satisfactory approach.

```
public void createBufferImage(int[][] redM, int[][] greenM, int[][] blueM)
  {
      //Normalize the matrix.  Remove the SIGN As well
      redMatrix   = cu.normilze(redM);
      greenMatrix = cu.normilze(greenM);
      blueMatrix  = cu.normilze(blueM);

      int red,green, blue;      //variables for RGB

      //Create buffer storage
      int[] data = new int[width*height];
      int arrayCounter =0;                 //Buffer index

      //First scan through each row in outer loop. height=512
      for (int y = 0; y < height; y++)
      {
       //First scan through each column in inner loop.width=614
        for (int x = 0; x < width; x++)
        {
          //Conver to 0-255 RGB color schem
          red   = (int) (redMatrix[y][x] );
          green = (int) (greenMatrix[y][x]);
          blue  = (int) (blueMatrix[y][x] );
          data[arrayCounter++] = ((red) <<16)| ((green&0xff)<<8)| (blue&0xff) ;
        }
      }
      image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
      image.setRGB(0,0,width, height,data,0,width);

      //Let show it
      setVisible(true);
  }

  public void paint(Graphics g)
  {
      g.drawImage(image,0,0,this);
  }
```

Figure 3.3. BufferedImage data buffer operations, as used in Viewer.java and produced for this thesis.

D. 3D VISUALIZATION

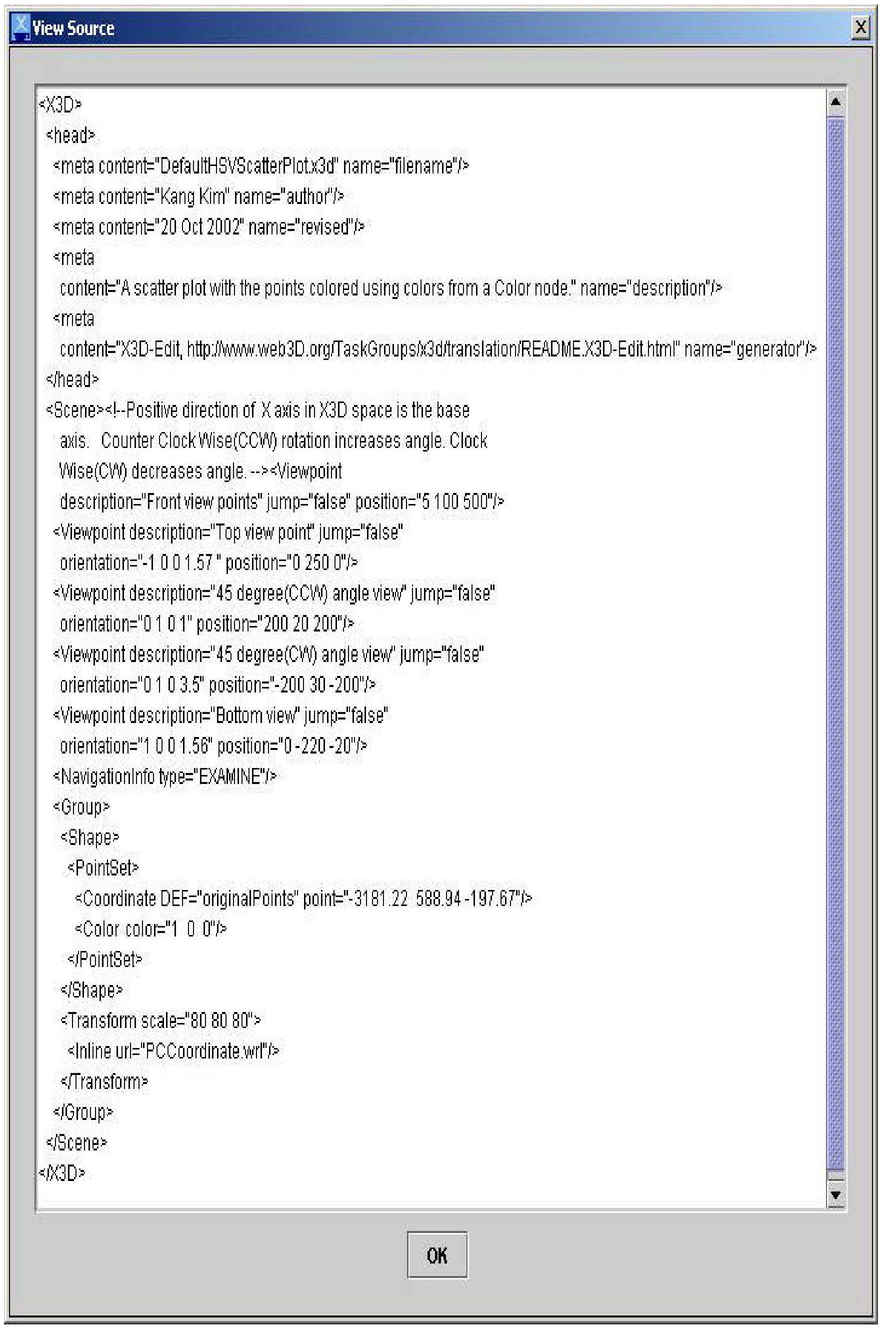
1. Process Overview

The 3D visualization goal is to view the conical HSV data space for the selected region of interest in the most intuitive possible manner, in order to assess the effectiveness of PCT mappings. The data parameters in 3D space are color attributes in the X3D Color node and the point attribute is in the X3D Coordinate nodes. Once a X3D scene with default point and color attribute values is loaded into memory, the processed data in the Java application is computed and then appended into the loaded X3D scene graph. This X3D scene is validated as syntactically correct by the X3D compact DTD. Once the composite data tree is formed, the x3dToVRML97 stylesheet is applied to transform X3D into VRML97 syntax. This VRML scene is loaded by a Xj3D or VRML capable browser for 3D viewing.

2. Extensible Markup Language (XML)

The explosion of the World Wide Web during the 1990's can be directly attributed to the creation and implementation of the Hypertext Markup Language (HTML). The Hypertext Markup Language is a subset of a more extensive language called the Standard Generalized Markup Language (SGML). HTML has enjoyed overwhelming success due to its simplicity. The Hypertext Markup Language's simplicity lies in its use of markup tags (character elements bracketed by '<' and '>') that are predefined by a standardized Document Type Definition (DTD). Essentially, HTML elements identify how web browsers display information, text and pictures.

While HTML's standardized DTD enables simplicity, it does not readily allow for the insertion of metadata (data describing data) within a web page. This lack of extensibility leads to unstructured data within HTML web pages. Consequently, it is often difficult to access and manipulate data written in HTML. Originally, this inability to incorporate metadata into information was solved somewhat by the predecessor Standard Generalized Markup Language (SGML). SGML permits users to define their own tagsets via a Document Type Definition (DTD). Users are thereby able to insert metadata through the creation of their own element types. However, SGML's element type advantages are still overshadowed by it cumbersome specifications (Khare, 1999). In an effort to combine the simplicity of HTML and the information exchange advantage of SGML, the World Wide Web Consortium (W3C) designed a new language called Extensible Markup Language (XML). XML is used to define other languages and custom data formats in a Web-compatible manner.



```
<X3D>
 <head>
  <meta content="DefaultHSVScatterPlot.x3d" name="filename"/>
  <meta content="Kang Kim" name="author"/>
  <meta content="20 Oct 2002" name="revised"/>
  <meta
   content="A scatter plot with the points colored using colors from a Color node." name="description"/>
  <meta
   content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html" name="generator"/>
 </head>
 <Scene><!--Positive direction of X axis in X3D space is the base
   axis.   Counter Clock Wise(CCW) rotation increases angle. Clock
   Wise(CW) decreases angle. --><Viewpoint
   description="Front view points" jump="false" position="5 100 500"/>
  <Viewpoint description="Top view point" jump="false"
   orientation="-1 0 0 1.57 " position="0 250 0"/>
  <Viewpoint description="45 degree(CCW) angle view" jump="false"
   orientation="0 1 0 1" position="200 20 200"/>
  <Viewpoint description="45 degree(CW) angle view" jump="false"
   orientation="0 1 0 3.5" position="-200 30 -200"/>
  <Viewpoint description="Bottom view" jump="false"
   orientation="1 0 0 1.56" position="0 -220 -20"/>
  <NavigationInfo type="EXAMINE"/>
  <Group>
   <Shape>
    <PointSet>
     <Coordinate DEF="originalPoints" point="-3181.22  588.94 -197.67"/>
     <Color  color="1  0  0"/>
    </PointSet>
   </Shape>
   <Transform scale="80 80 80">
    <Inline url="PCCoordinate.wrl"/>
   </Transform>
  </Group>
 </Scene>
</X3D>
```

Figure 3.4. XML view for the DefaultHSVScatterPlot scene from X3D-Edit.

```
<!ELEMENT Color (IS?) >
<!ATTLIST Color
      color       %MFColor; #IMPLIED
      containerField NMTOKEN "color"
      class       CDATA       #IMPLIED
      DEF         ID        #IMPLIED
      USE         IDREF     #IMPLIED>

.
.
.
.
.
.
.
<!ELEMENT Coordinate (IS?) >
<!ATTLIST Coordinate
      point        %MFVec3f; #IMPLIED
      containerField NMTOKEN "coordinate"
      class       CDATA       #IMPLIED
      DEF          ID        #IMPLIED
      USE          IDREF     #IMPLIED>
```

Figure 3.5. Color and Coordinate node definition excerpted from x3d-compact.dtd used in this thesis.

3. Document Type Definition (DTD)

A Document Type Definition (DTD) collects a list of tags that defines specific elements and attributes. DTDs also describe the relationships and format between elements and attributes (Harold, 2000). DTDs ensure the validity of an XML document by requiring the data in the XML document to adhere to the prescribed format and structure. DTDs are enablers of interoperability because they can serve as the information exchange standard for unrelated organizations using XML applications. In this study, tx3d-compact.dtd was used to provide X3D scene validation. Figure 3.5 shows one portion of x3d-compact.dtd DTD content.

4. XML Schema

The XML Schema provides alternative to DTDs introduced in the previous section for validating XML documents. Like DTD's, schemas must be used with validating parsers. Schemas are expected to replace DTD's as the primary means of describing the document structure (Deitel, 2002). Unlike DTDs, the schema does not use the SGML-inspired grammar. Instead, the schema uses a carefully defined XML syntax.

Since schemas are themselves XML documents, they can be manipulated, i.e., elements added, elements removed, etc., as in any other XML document.

5. Virtual Reality Modeling Language (VRML)

The Virtual Reality Modeling Language (VRML) is used to define interoperable content for three-dimensional (3D) virtual worlds. One key feature of VRML is that it is an ISO standard designed to be used over the World Wide Web in a browser environment. The various examples explored in this thesis are encoded in X3D, which is the third-generation successor to VRML. X3D includes both XML and VRML-style data encodings.

The fundamental design structure in VRML is a scene graph. A 3D scene graph is composed in VRML by grouping and encoding content via nodes. These nodes are then used to display objects such as primitive shapes (such as Box), elevation grids or complex indexed face sets. The nodes also specify groupings of sub-nodes and can indicate appearance, interaction and movement of events throughout the scene graph.

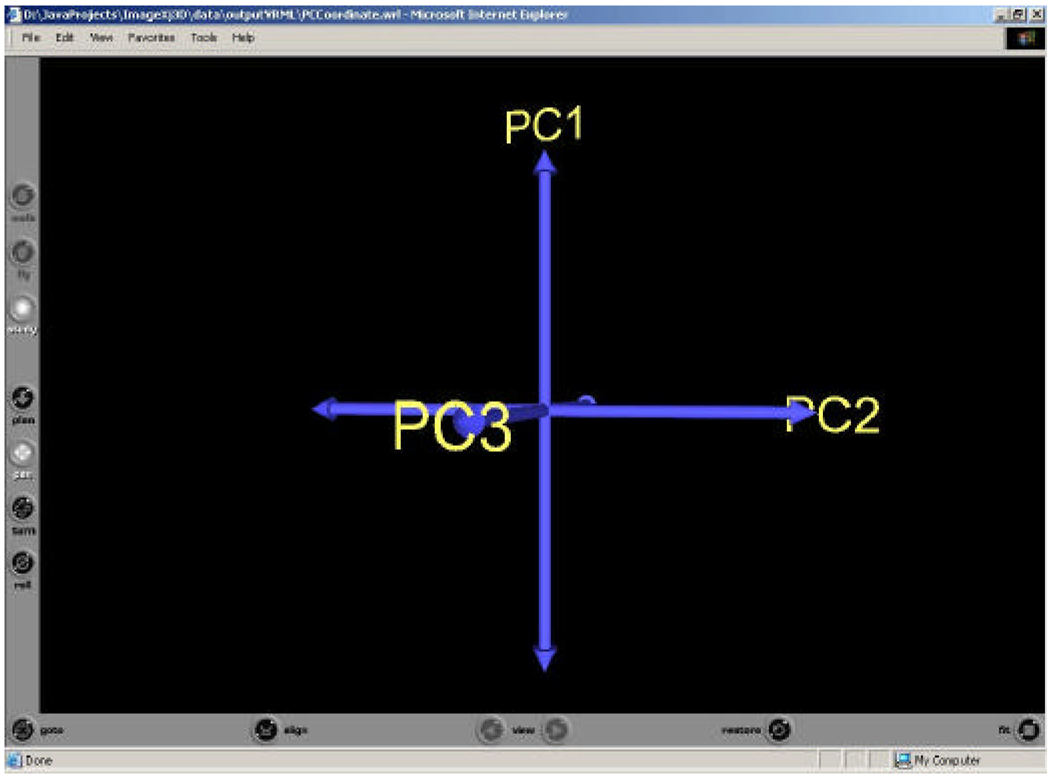

D:\JavaProjects\ImageJ3D\data\outputVRML\PCCoordinate.wrl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
PC1
PC2
PC3
goto
align
view
restore
fit
Done
My Computer

Figure 3.6. VRML example indicating Principal Component (PC) space axes.

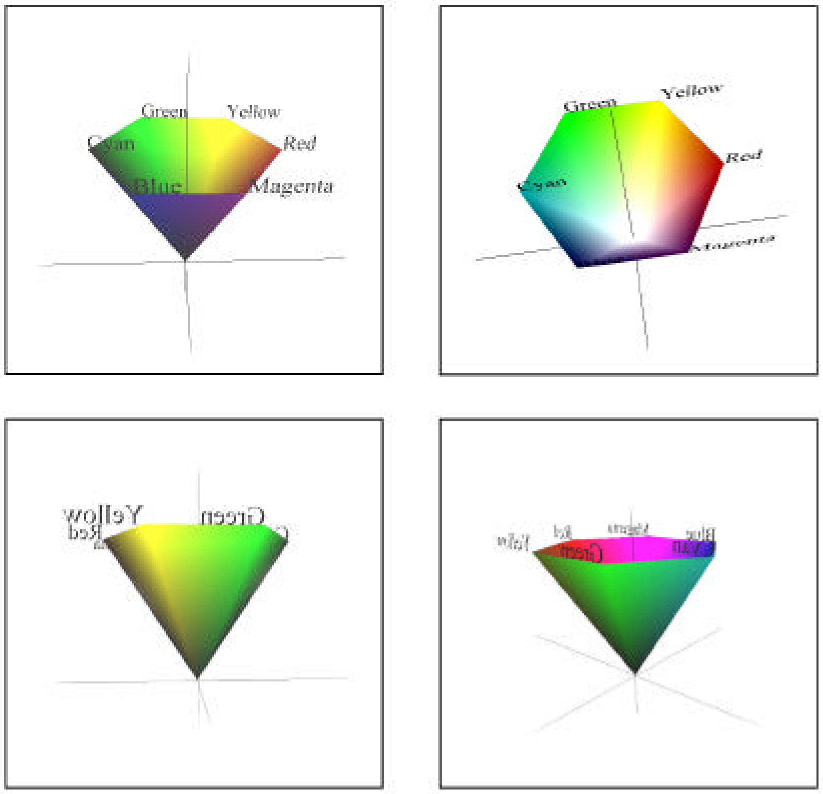
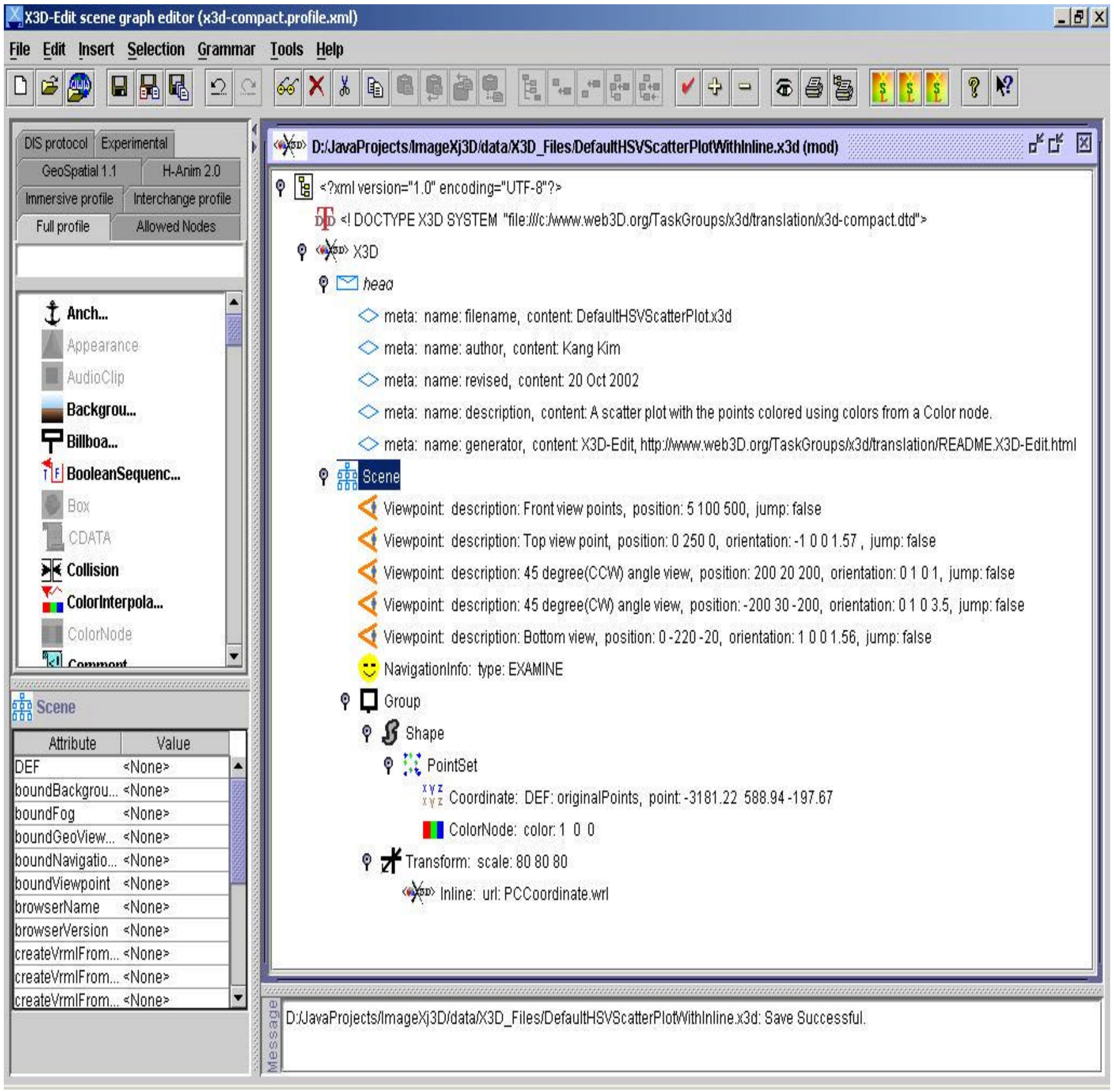


Figure 3.7. HSV color space model by X3D-Edit.

6. Extensible 3D (X3D) Graphics Language

The next-generation VRML specification is known as Extensible 3D Graphics (X3D) (Brutzman). X3D is more than an update to VRML97. It is a redesign of the encoding and underlying code structure by employing XML. The new X3D standard constructs a DTD target that allows users to develop well-formed and validated scene graphs. Using XML provides X3D with a robust structure and extensibility. Extensible 3D retrieves the same fundamental structure as the VRML97 standard and is a superset of functionality that is fully backward compatible.

Using an X3D software development kit and the X3D-Edit authoring tool, developers can produce validated scene graphs with error-free editing (Extensible 3D Task Group, 2000). The X3D-Edit utilizes IBM’s Xeena XML editor that has been configured to facilitate straightforward development of scene graphs that conform to the X3D DTD. The X3D-Edit tool converts X3D documents to VRML97 via a XSL stylesheet and then automatically launches a browser for convenient viewing. Figure 3.8 shows a screen capture of the X3D-Edit. (Extensible 3D Task Group, 2000)



D:/JavaProjects/ImageXj3D/data/X3D_Files/DefaultHSVScatterPlotWithInline.x3d (mod)

```
<?xml version="1.0" encoding="UTF-8"?>
  <! DOCTYPE X3D SYSTEM "file:///c:/www.web3D.org/TaskGroups/x3d/translation/x3d-compact.dtd">
  X3D
    head
      meta: name: filename, content: DefaultHSVScatterPlot.x3d
      meta: name: author, content: Kang Kim
      meta: name: revised, content: 20 Oct 2002
      meta: name: description, content: A scatter plot with the points colored using colors from a Color node.
      meta: name: generator, content: X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html
    Scene
      Viewpoint: description: Front view points, position: 5 100 500, jump: false
      Viewpoint: description: Top view point, position: 0 250 0, orientation: -1 0 0 1.57 , jump: false
      Viewpoint: description: 45 degree(CCW) angle view, position: 200 20 200, orientation: 0 1 0 1, jump: false
      Viewpoint: description: 45 degree(CW) angle view, position: -200 30 -200, orientation: 0 1 0 3.5, jump: false
      Viewpoint: description: Bottom view, position: 0 -220 -20, orientation: 1 0 0 1.56, jump: false
      NavigationInfo: type: EXAMINE
      Group
        Shape
          PointSet
            Coordinate: DEF: originalPoints, point: -3181.22 588.94 -197.67
            ColorNode: color: 1 0 0
        Transform: scale: 80 80 80
          Inline: url: PCCoordinate.wrl
```

Attribute	Value
DEF	<None>
boundBackgrou...	<None>
boundFog	<None>
boundGeoView...	<None>
boundNavigatio...	<None>
boundViewpoint	<None>
browserName	<None>
browserVersion	<None>
createVrmlFrom...	<None>
createVrmlFrom...	<None>
createVrmlFrom...	<None>

D:/JavaProjects/ImageXj3D/data/X3D_Files/DefaultHSVScatterPlotWithInline.x3d: Save Successful.

Figure 3.8. X3D-Edit Authoring Tool Example. This is the basis to generate the default XML scene and check its validity.

X3D provides the critical link between XML documents and the virtual worlds of this thesis. Although VRML97 is the basis for many of the models developed, X3D provides the structure and flexibility to transform XML documents into valid scene graphs.

7. Xj3D

In this thesis, the Xj3D Player provides one way to view the X3D graphic scene after the scene is manipulated by the Java application. Xj3D is the open-source rendering API and toolkit developed by Yumetech and others (Xj3D Task Group, 2002). Xj3D is a project of the Web 3D Consortium that creates a toolkit for VRML97 and X3D content written completely in Java. This toolkit may be used to the import VRML content into a custom application, or to create a fully-fledged browser. The initial impetus for this project was to create a file loader for the Java3D API and started with a grant of code from Sun Microsystems to W3C. Over time, the project has grown in requirements and now encompasses many other features as well. For example, it is being used as one of the main testing grounds to verify the work on the new X3D specification. It currently uses Java3D as the scene-rendering engine (Xj3D Task Group, 2002).

.

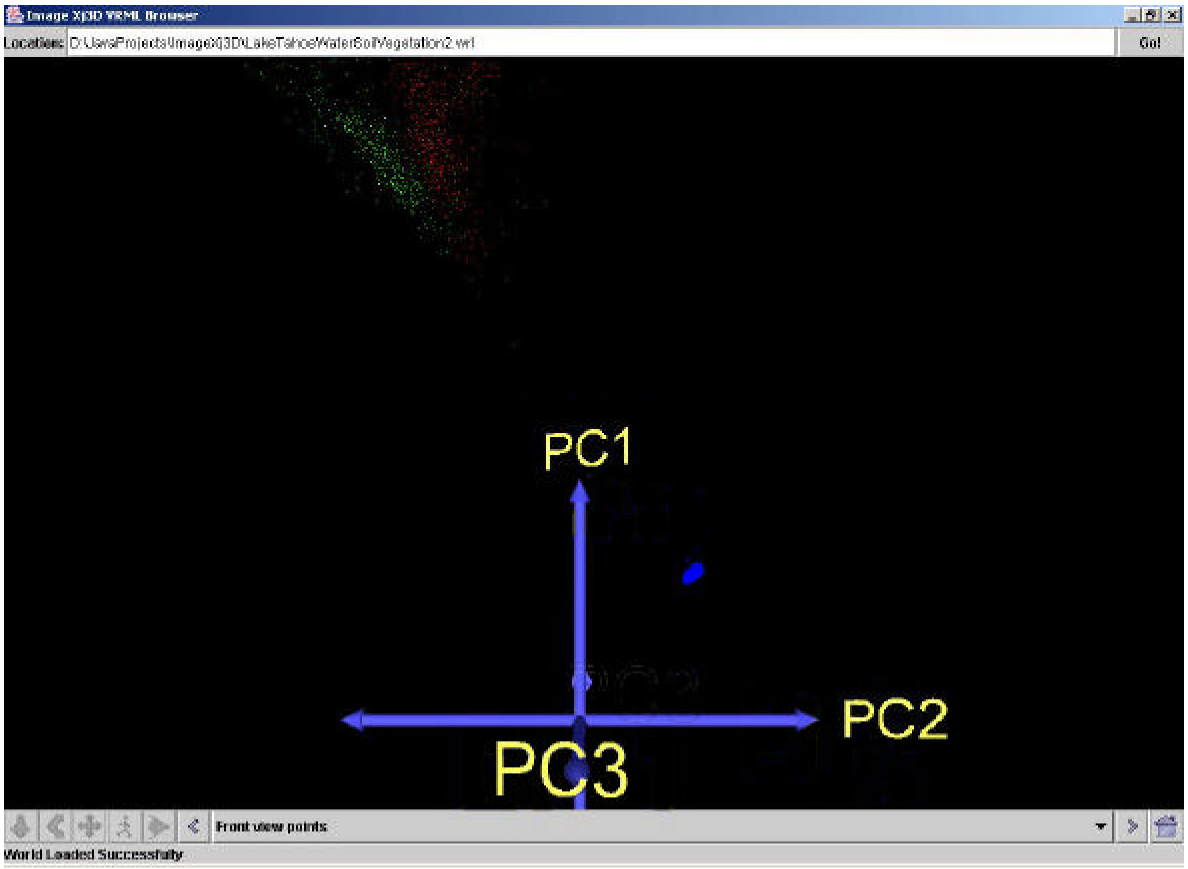


Image Xj3D VRML Browser
Location: D:\JavaProjects\ImageXj3D\LakeTahoeWaterSoilVegetation2.wrl
Go!
PC1
PC2
PC3
Front view points
World Loaded Successfully

Figure 3.9. Simulation result viewing with Xj3D.

E. SUMMARY

This chapter details the high-level concepts of the many technologies explored and exploited to provide a contextual understanding of the remainder of the thesis. If more information is required for a more in-depth understanding, the reader is referred to the List of References as well as those mentioned above.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROBLEM DEFINITION AND ANALYTIC APPROACH

A. INTRODUCTION

This chapter provides the insight into the subject of this thesis. The first section introduces the mapping strategies currently available. The second section describes the Principal Components (PC) mapping strategy in depth, as related to Tyo's invariant-display strategy. The last section presents new techniques to improve this proposed strategy, such as viewing the conical data space of Principal Component (PC) space in 3D, developing a global statistic and viewing the resultant 2D image of RGB transformation from PC data.

B. MAPPING STRATEGIES OVERVIEW

The two most common approaches to mapping HSI data into pseudocolor involve mapping band data or principal components data at each pixel into an RGB triple. In the former strategy, a user selects three HSI bands in the hopes of capturing either large-scale image variations or particular spectral features. An example of such a mapping is presenting a long-, mid-, and short wavelength visible bands as an RGB image. This approximates what a human observer might see if actually looking at the scene. An example of this mapping strategy is shown in Figure 4.1.B. This class of mapping strategy can be very powerful, as highly specialized colormaps can be designed that are tailored to particular applications, such as locating a specific spectral feature throughout a scene. However, there are some drawbacks associated with this strategy. First, any spectral feature that does not overlap with chosen bands will not be represented. Second, there is no *a priori* way to predict the color representation of specific objects in the final images, or to precisely interpret them.

A second strategy attempts to solve some of those drawbacks. It involves taking a Principal Components transformation (PCT) and mapping the resulting PC images into an RGB triple. Examples of such images are presented in Figure 4.1.

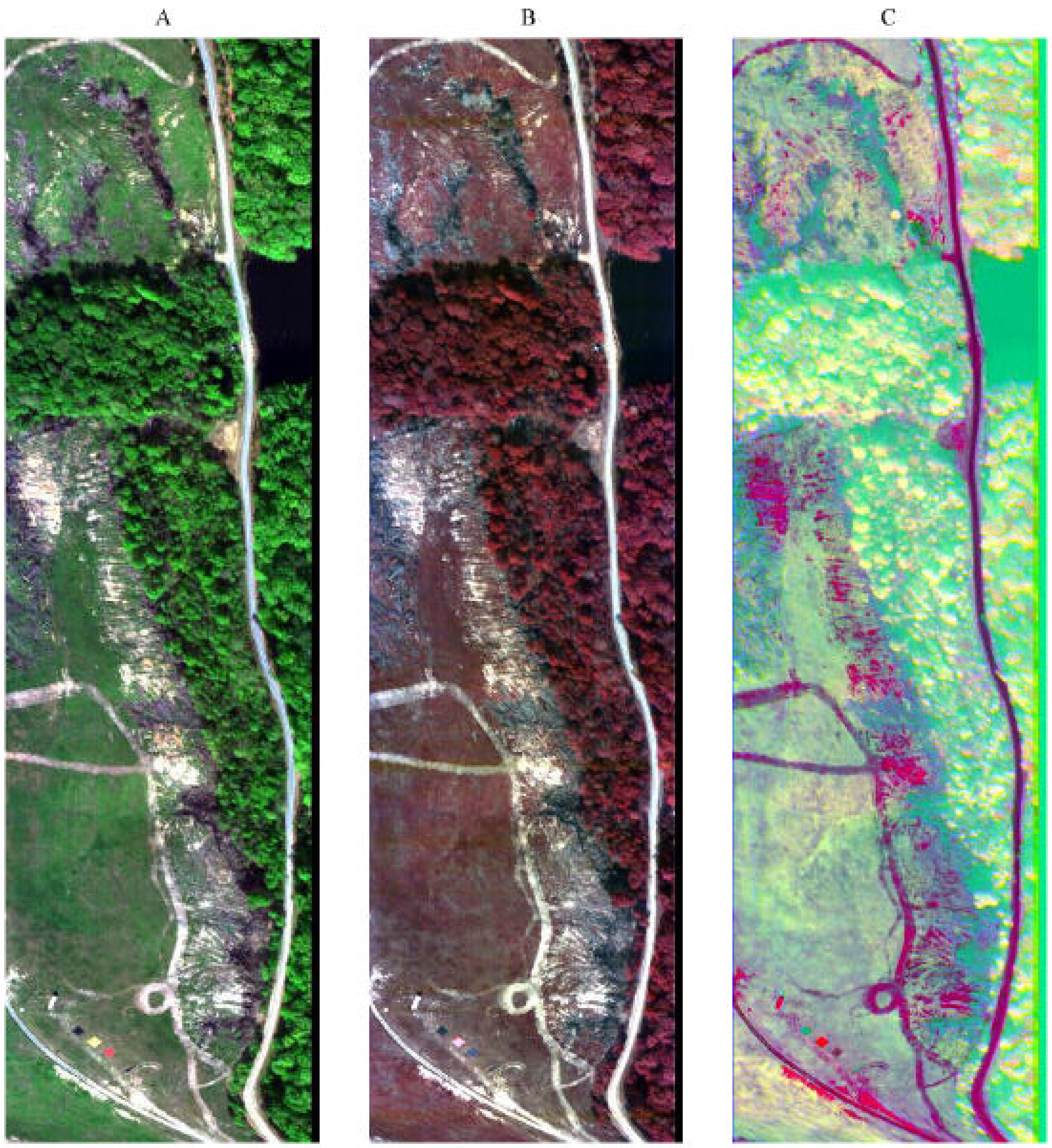


Figure 4.1. Three common three-color mapping strategies from Night Vision Imaging System (NVIS) data. A. Natural color image. This is an approximation of what a human observer would see in the image maps bands at 630nm, 550nm, and 450nm into R, G, and B. B. Typical false color in Infrared. The second image presents bands at 600nm, 1000nm, and 2000nm. C. Mapping PC images into RGB Triples. 1st PC mapped into red, 2nd PC into green, and 3rd into blue.

The PC images are formed by diagonalizing the covariance matrix of the data, and projecting the HSI scene onto the resulting (orthogonal) eigenvectors. The eigenvectors are linear combinations of variables, which include the spectral features that contribute most to the scene variance. They represent statistically uncorrelated channels and are ordered in decreasing amounts of scene variance. The derived variables provide sample information from across the scene, and reduce the chances that an important spectral feature, from a variance standpoint, will be missed in the final representation. The problem with the mapping in Figure 4.1 is that they map orthogonal data (PC images) into non-orthogonal display channels (RGB intensities). The result is an image that is often difficult to interpret.

Recently, a PC-based strategy was presented that sought to eliminate the problems associated with the above mapping. That strategy exploits the similarities between HSI data and human color vision. The orthogonal PC channels are mapped into orthogonal display channels in an ergonomic manner. The end goal is to derive an invariant mapping strategy for HSI data that consistently and intuitively represents important scene constituents in the final scene. Advanced processing methods can simultaneously be applied to the HSI data, enabling overlay of identification information.

C. PRINCIPAL COMPONENT (PC) MAPPING STRATEGY REVIEW

1. Overview

The close dependency between effective interpretation of HSI data and human color vision is well known (Dierson, 2000). A PC analysis of the human photoreceptor spectral response produces three statically orthogonal channels. One channel is roughly achromatic, one channel demonstrates a difference between long- and mid- wavelength spectral content (red-green), and a third channel is trimodal, and nominally represents the difference between short- and mid wavelength information (blue-yellow). These three orthogonal dimensions can be used to define orthogonal directions in a 3-dimensional, conical data space. The three HSV coordinates--hue (angle within the R-G/B-Y plane), saturation (radius in the R-G/B-Y plane divided by total intensity), and value (total intensity)--define a location within the HSV cone and are commonly used for constructing pseudocolor images.

Analysis of HSI data yields a first PC that is usually related to the average scene illumination. That channel typically has slow spectral variation, except in atmospheric absorption bands, and closely resembles the solar distribution convolved with the atmospheric transmission. When an image is dominated by a particular material, that spectral signature might also contribute significantly to the scene variance and may show up in the first PC as well. The first PC image contains a basic panchromatic intensity image and has much of the high spatial-frequency variations. The higher PCs also have lower-spatial frequency information that is typically associated with regions of like spectral content. The close analogy between the HSI and color vision analyses leads to the proposed mapping information often associated with geography. Successively higher PCs tend to have a more rapid spectral strategy. Equation Section 4

$$\begin{aligned} f &= \arctan\left(\frac{P_3}{P_2}\right) \to H \\ r &= \frac{\sqrt{P_2^{\,2} + P_3^{\,2}}}{P_1} \to S \\ P_1 &\to V \end{aligned} \tag{4.1}$$

where P_1, P_2, and P_3 are the 1st, 2nd, and 3rd PC values, f and r are angle within the R-G/B-Y plane and radius in the R-G/B-Y plane divided by total intensity, and H, S, and V are the hue, saturation, and value. The basic properties of this transformation are discussed (Tyo et al., 2001).

2. Analysis

The strategy outlined above has several advantages and disadvantages that are worth mentioning. First, the mapping capitalizes on similarities between the PC channels in HSI and color vision to create an ergonomic strategy that preserves orthogonality relationships in the final mapping. The spatial frequency structure of the PC images in HSI nicely matches the spatial sensitivity of the corresponding color channels. This match points towards a possible compression strategy, namely a hybrid spatial/spectral compression scheme, that uses information about the spatial frequency structure of higher PC channels to reduce noisy and/or less important data from a scene. This is the strategy utilized by human vision. High spatial frequency information that is presented in the (R-G) and (B-Y) channels is not readily perceived, and helps to minimize the visual bandwidth necessary to process color scene.

D. INVARIANT DISPLAY STRATEGY

The long-term goal of this project is the development of an invariant display strategy that can be broadly applied to HSI data. The direct implementation of Equation 4.1 is not an invariant strategy, since the PCs are calculated from in scene statistics. Furthermore, implementation of Equation 4.1 does not guarantee that materials are going to be presented in hues that are intuitive to the observer. Below are two aspects of this strategy that can then be considered for further improvement.

1. Conical Data Shape

The transformation in Equation 4.1 projects the high-dimensional HSI data into a 3D conical space. That space is tightly bunched about the P_1 axis, since the variance associated with first PC is often in excess of 90% of the total scene variance. The effect of this transformation on HSI data is shown in the next chapter.

2. Post-Rotation of RGB Data

Sometimes it may be desired to use in-scene statistics to compute the eigenvectors for transformation. These eigenvectors give the best ordering of the in-scene variance, and highlight the most important features of the particular image. However, use of in-scene statistics can lead to colormaps that are not always intuitive. A method for overcoming this difficulty is proposed in this section.

When the spectral data includes bands in the visible, a 3-color composite image can be constructed that closely mimics what a human observer would see. In Figure 4.1, an important scene constituent is the grass. The color transformation can be rotated to ensure that the materials that closely resemble grass are presented with a particular hue, i.e., the color green. The mapping was computed using such a post-rotation. The second and third eigenvectors were still obtained using in-scene data. If a strategy is developed to arrive at global eigenvectors, then Equation 4.1 can be appropriately modified to ensure that important materials are presented in a visual standard form for improved recognizability.

E. SUMMARY

In this chapter, the fundamental thesis problem was restated and analyzed to identify the necessary tasks for using 3D visualization techniques to improve PCT mappings of particular interest: comparing conical data shape in PC space with HSV color space and post-rotating RGB transformation. In the following chapters, previous work is extended to examine the conical nature of HSI data in the PC-based 3D space.

V. VISUALIZATION IMPLEMENTATION

A. INTRODUCTION

The previous chapter demonstrated that two aspects for this study need to be investigated: the 3D view of Principal Component (PC) data from selected Regions Of Interest (ROI) and the resulting image presented in RGB. This chapter focuses on how the 3D view of the HSV mapping is generated through the creation of a Java application and X3D scene, in order to achieve the first objective. The next visualization section presents what data and graphics parameters need to be manipulated. The following section describes the overall application architecture. The final section describes how the X3D scene is generated.

B. VISUALIZATION CONSIDERATIONS

This section explains what data parameters are used to visualize, what graphical parameters are available in X3D, and how to map data parameters to graphics parameters. Figure 5.1 shows the entire process to generate graphics parameter from image data parameters.

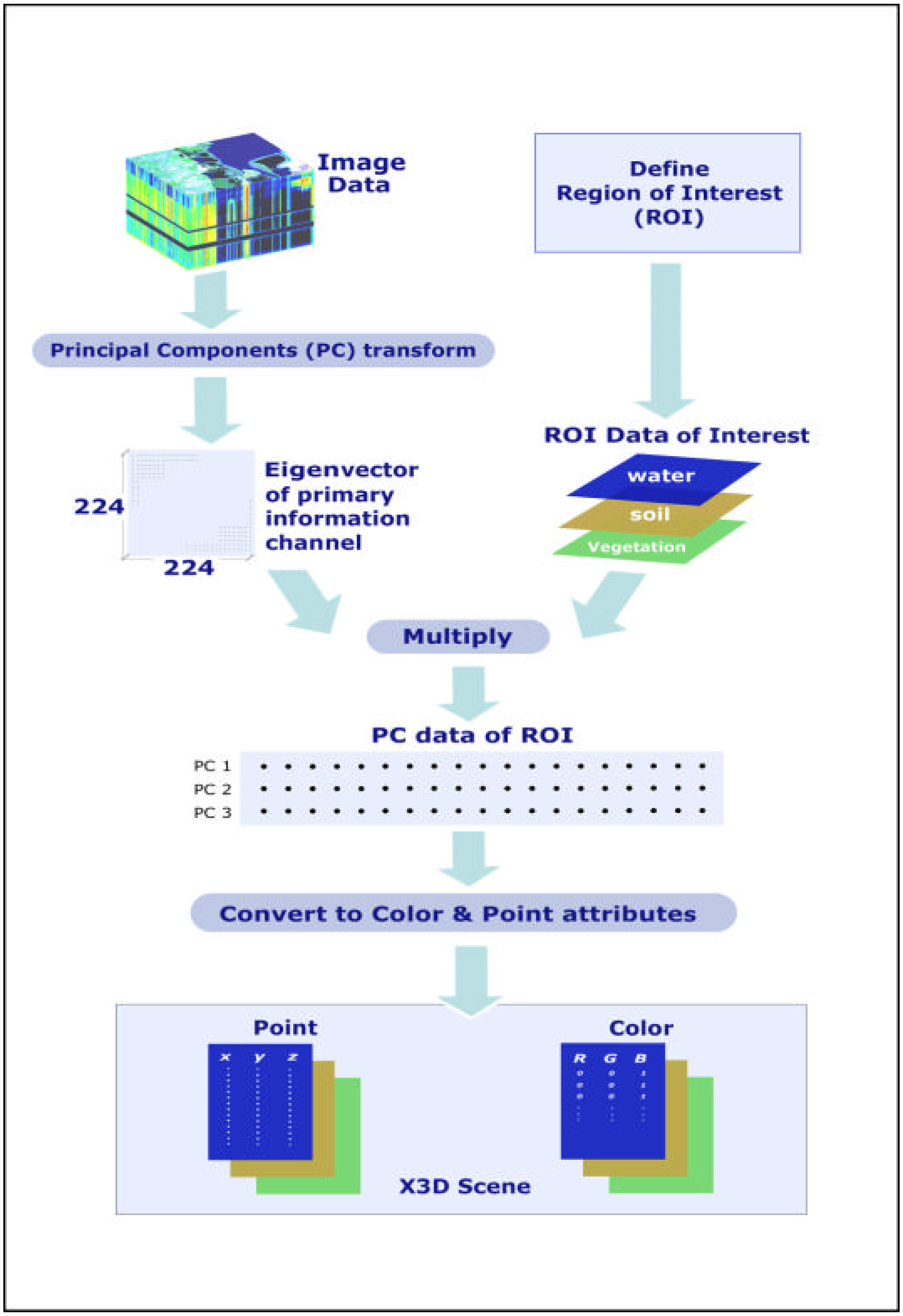


Image Data
Define Region of Interest (ROI)
Principal Components (PC) transform
ROI Data of Interest
water
soil
Vegetation
224
224
Eigenvector of primary information channel
Multiply
PC data of ROI
PC 1
PC 2
PC 3
Convert to Color & Point attributes
Point
x y z
Color
R G B
X3D Scene

Figure 5.1. Data process flow for converting and mapping HSI data.

1. Data Parameters

There are two data parameters needed to project the reduced dimensionality of Principal Component (PC) results into a 3D display space: the bcation of each point in PC space, and the color of points according to the material classification to which the ROI belongs. Referring to Figure 5.1, Hyperspectral Imagery (HSI) data for Regions of Interest (ROI) parameters are converted to PC space by application of an eigenvector rotation. An initial HSI image has three dimensions for width of image, height of image and the number of available bands. For instance, Lake Tahoe HSI data have the dimensions 614 by 512 by 224. This 3D array is transformed via Principal Components (PC), yielding a new 3D array of the same size, keeping only the first three bands in the new color space result in a reduced dimensionality, here, by example, 614 by 512 by 3 (See Richards for calculation detail). The rotation is defined by the In-Scene statistics eigenvectors. Conceptually, the ROI data are transformed through the same matrix rotation, as indicated in parallel in Figure 5.1. In practice, the ROI could be extracted from the transformed Image data at a later stage. ROI are defined for this work interactively by using Environment for Visualizing Image (ENVI) software from Research System Inc. as shown in Figure 5.2. Once the ROI is defined, the data subset can also be transformed into the three dimensions defined by the first three PCs.

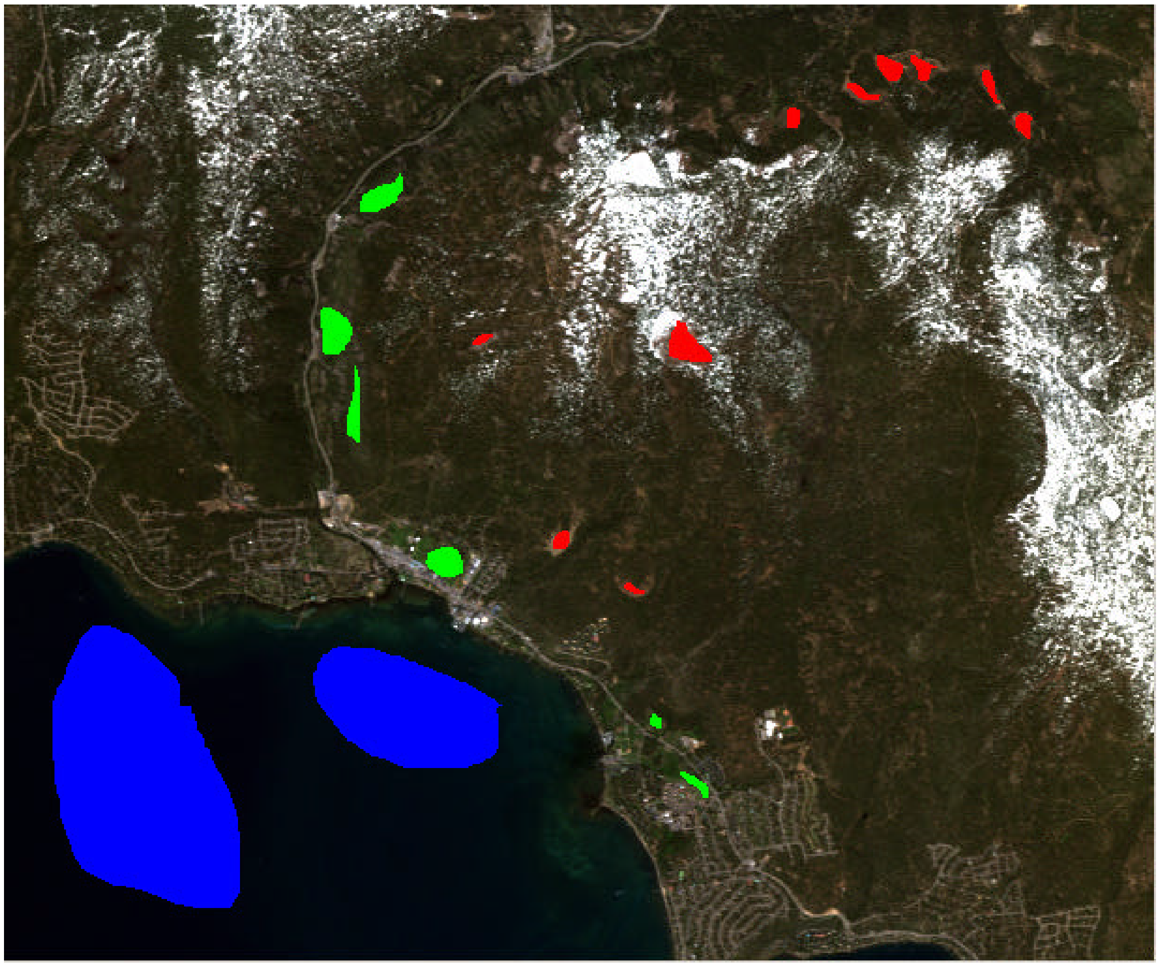
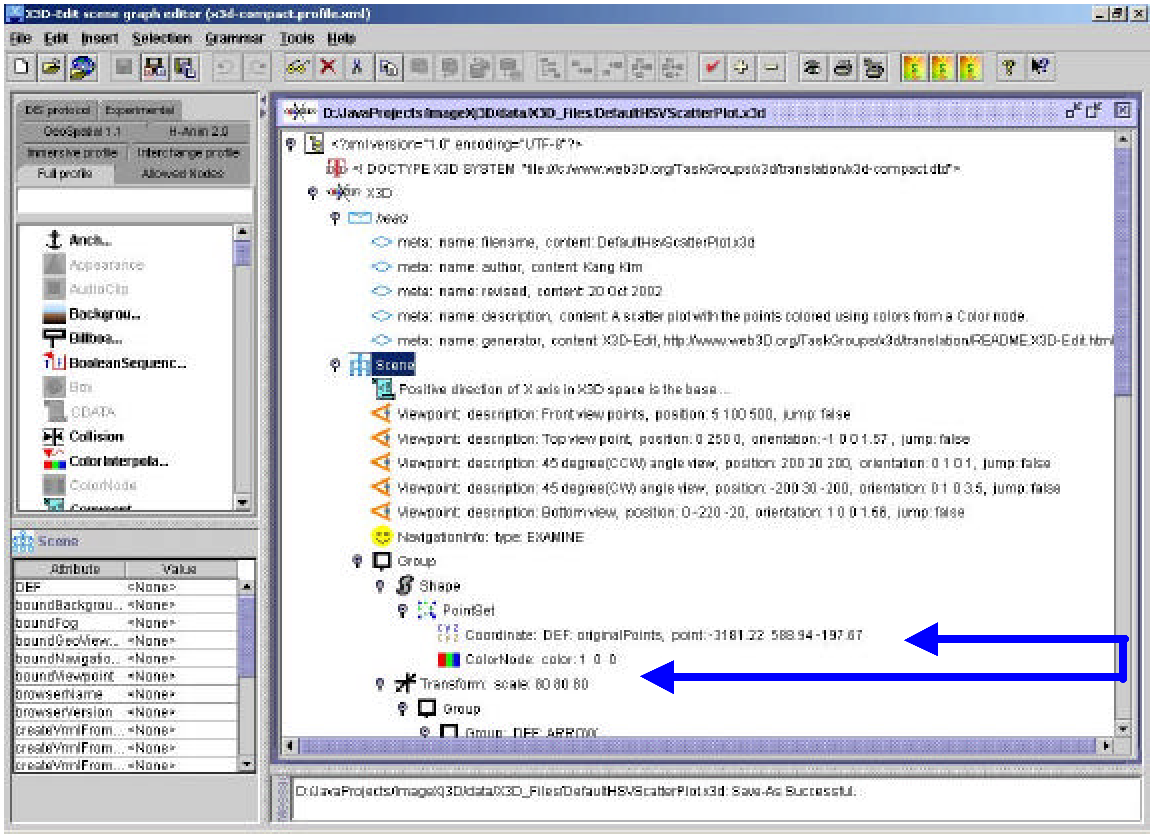


Figure 5.2. Region of Interest (ROI) for vegetation in green polygon, water in blue, and soil in red in Lake Tahoe scene with true color bands. The total number of selected points is 11059.

2. Mapping Converted Data to X3D

In X3D, the Coordinate and Color nodes contain point and color data for a PointSet. Each attribute, respectively, contains the x y z location and R G B triple. Thus, data parameters can be directly mapped to graphics parameters. Figure 5.3 shows these two attributes as used in the default X3D file of this thesis. The PC1, PC2, PC3 axes provide the location in PC space (Tyo et al., 2001) and material classification is mapped to the color attribute in the Color node in X3D. For instance, assume the first point of ROI is vegetation and its PC coordinates turn out to be –3181.22, 588.94, and –194.67 as shown in Figure 5.3. as indicated by the arrows. The color of this point is mapped to the green color as (0, 1, 0) for RGB triple, since this point is classified as vegetation. This entire PC data cloud is shown in 3D using the HSV color space.



Attribute	Value
DEF	<None>
boundBackgrou...	<None>
boundFog	<None>
boundGeoView...	<None>
boundNavigatio...	<None>
boundViewpoint	<None>
browserName	<None>
browserVersion	<None>
createVrmlFrom...	<None>
createVrmlFrom...	<None>
createVrmlFrom...	<None>

```
D:\JavaProjects\imageX3D\data\X3D_Files\DefaultHSVScatterPlot.x3d
<?xml version="1.0" encoding="UTF-8"?>
<! DOCTYPE X3D SYSTEM "file://c:/www.web3D.org/TaskGroups/x3d/translation/x3d-compact.dtd">
  X3D
    head
      meta: name: filename, content: DefaultHsvScatterPlot.x3d
      meta: name: author, content: Kang Kim
      meta: name: revised, content: 20 Oct 2002
      meta: name: description, content: A scatter plot with the points colored using colors from a Color node.
      meta: name: generator, content: X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html
    Scene
      Positive direction of X axis in X3D space is the base ...
      Viewpoint: description: Front view points, position: 5 100 500, jump: false
      Viewpoint: description: Top view point, position: 0 250 0, orientation: -1 0 0 1.57 , jump: false
      Viewpoint: description: 45 degree(CCW) angle view, position: 200 30 200, orientation: 0 1 0 1, jump: false
      Viewpoint: description: 45 degree(CW) angle view, position: -200 30 -200, orientation: 0 1 0 3.5, jump: false
      Viewpoint: description: Bottom view, position: 0 -220 -20, orientation: 1 0 0 1.66, jump: false
      NavigationInfo: type: EXAMINE
      Group
        Shape
          PointSet
            Coordinate: DEF: originalPoints, point: -3181.22 588.94 -197.67
            ColorNode: color: 1 0 0
        Transform: scale: 80 80 80
          Group
            Group: DEF: ARROW

D:\JavaProjects\imageX3D\data\X3D_Files\DefaultHSVScatterPlot.x3d: Save-As Successful.
```

Figure 5.3. The content of DefultHsvScatterPlot.x3d used in this thesis.

C. APPLICATION DESIGN

This section presents what design concepts have been applied in building the application, and describes the overall application structure of each software package. Figure 5.1 shows the Universal Modeling Language (UML) diagram for each package.

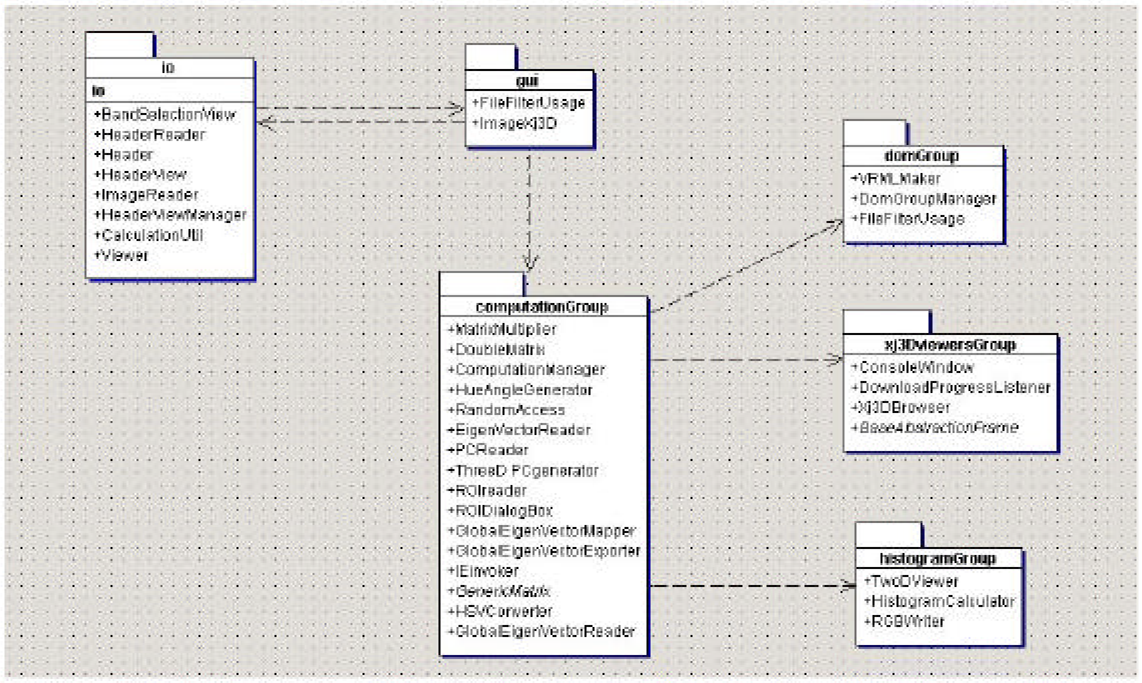


Figure 5.4. Universal Modeling Language (UML) diagram of the entire Java application by each package. UML is generated by TogetherSoft software.

1. Software Design Patterns Overview

Design patterns describe standard ways to correctly perform common programming tasks. Many different software design pattern concepts exist: Creational pattern, Behavior pattern, Structural pattern, and System pattern (Stelting, 2002). Among these concepts, a creational pattern has been applied to develop this application, since it supports one of the most common tasks in Object-Oriented (OO) programming and an application of this complexity requires many objects to be instantiated over time. For instance, each package of this application consists of several classes. The result of the computation of each package is shared among the others.

2. Application Structure

This section provides a brief overview of the developed application in this thesis by each package. The application consists of six packages. Each package provides a different functionality as follows: loading image data, computing data, displaying resultant image, and viewing 3D Principal Component (PC) data space.

The application first loads the binary image data into memory. Once data is loaded, it is processed and calculated. Then, the application generates two types of datasets corresponding to the 3D projection of HSV mapped eigenvectors for the ROI, and also, the 2D resultant image.

a. GUI Package

Referring to Figure 5.4, this package is the topmost main package to control the rest of packages. This package provides the dialog boxes shown in Figure 5.5 that select the image to load. The image data itself is binary data with a 16-bit Signed Integer type in Band Sequential (BSQ) format as mentioned in Chapter II. This data does not have any precalculated metadata information of information, such as image width, image height, number bands, byte order or data type. The Header file in ASCII text format provides such information. It contains all the necessary information as shown in Figure 5.6. This header file information also can be shared with ENVI utilized above for defining ROI. Figure 5.7 shows the information of a Header file loaded into memory as well.

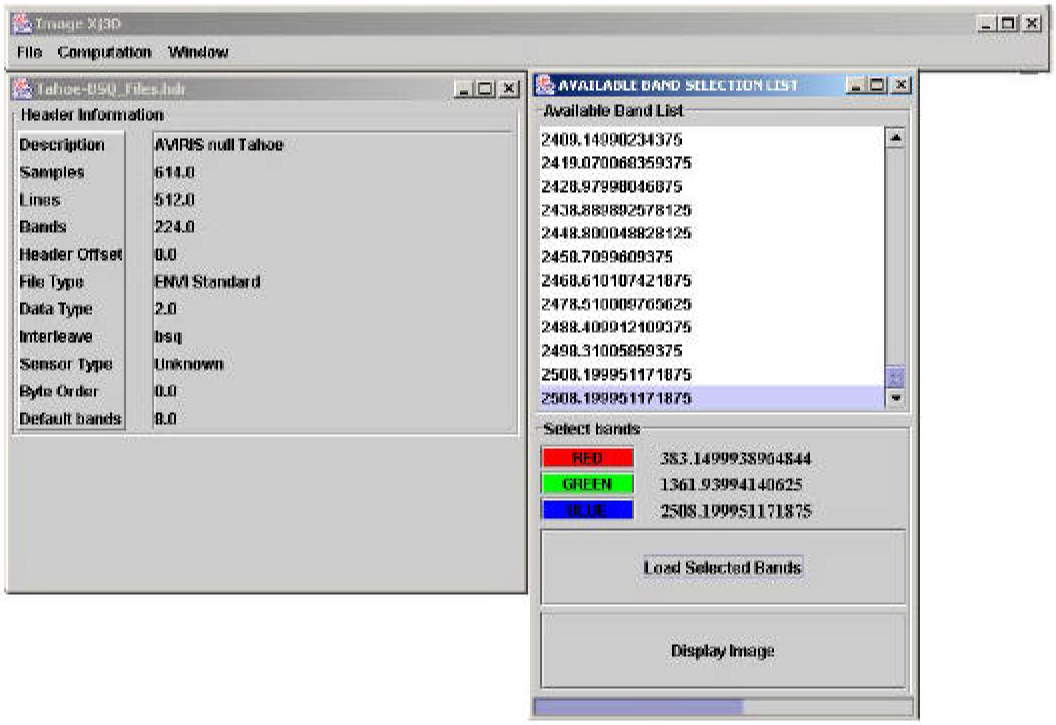
Figure 5.5. Entering Graphical User Interface (GUI) for GUI package.

```
description = {
  AVIRIS - Tahoe [Wed Sep 18 05:53:12 2002]}
samples = 614
lines   = 512
bands   = 224
header offset = 0
file type = ENVI Standard
data type = 2
interleave = bsq
sensor type = Unknown
byte order = 0
default bands = {28,18,8}
wavelength = {
  383.149994,  392.839996,  402.540009,  412.250000,  421.980011,  431.709991,
  441.459991,  451.220001,  460.989990,  470.760010,  480.549988,  490.339996,
  500.140015,  509.950012,  519.760010,  529.580017,  539.400024,  549.229980,
  559.070007,  568.900024,  578.739990,  588.580017,  598.429993,  608.270020,
  618.109985,  627.960022,  637.799988,  647.650024,  657.489990,  667.330017,
  677.169983,  687.000000,  664.299988,  673.869995,  683.440002,  693.020020,
  702.590027,  712.169983,  721.750000,  731.340027,  740.919983,  750.510010,
  760.090027,  769.679993,  779.270020,  788.869995,  798.460022,  808.049988,
  817.650024,  827.250000,  836.849976,  846.450012,  856.049988,  865.650024,
  875.250000,  884.849976,  894.460022,  904.059998,  913.669983,  923.270020,
  932.880005,  942.489990,  952.090027,  961.700012,  971.309998,  980.919983,
  990.530029, 1000.130005, 1009.739990, 1019.349976, 1028.959961, 1038.569946,
 1048.180054, 1057.790039, 1067.390015, 1077.000000, 1086.609985, 1096.209961,
 1105.819946, 1115.430054, 1125.030029, 1134.630005, 1144.239990, 1153.839966,
 1163.439941, 1173.040039, 1182.640015, 1192.239990, 1201.839966, 1211.430054,
 1221.030029, 1230.619995, 1240.219971, 1249.810059, 1259.400024, 1268.989990,
 1252.359985, 1262.319946, 1272.280029, 1282.250000, 1292.209961, 1302.170044,
 1312.130005, 1322.099976, 1332.060059, 1342.020020, 1351.979980, 1361.939941,
 1371.900024, 1381.859985, 1391.819946, 1401.780029, 1411.739990, 1421.699951,
 1431.660034, 1441.619995, 1451.579956, 1461.540039, 1471.500000, 1481.449951,
 1491.410034, 1501.369995, 1511.329956, 1521.280029, 1531.239990, 1541.199951,
 1551.150024, 1561.109985, 1571.060059, 1581.020020, 1590.969971, 1600.930054,
 1610.880005, 1620.839966, 1630.790039, 1640.739990, 1650.699951, 1660.650024,
 1670.599976, 1680.560059, 1690.510010, 1700.459961, 1710.410034, 1720.359985,
 1730.310059, 1740.270020, 1750.219971, 1760.170044, 1770.119995, 1780.069946,
 1790.020020, 1799.969971, 1809.920044, 1819.859985, 1829.810059, 1839.760010,
 1849.709961, 1859.660034, 1869.599976, 1879.550049, 1879.900024, 1889.949951,
 1900.000000, 1910.050049, 1920.089966, 1930.140015, 1940.180054, 1950.209961,
 1960.250000, 1970.280029, 1980.310059, 1990.339966, 2000.359985, 2010.380005,
 2020.400024, 2030.420044, 2040.430054, 2050.449951, 2060.449951, 2070.459961,
 2080.469971, 2090.469971, 2100.469971, 2110.459961, 2120.459961, 2130.449951,
 2140.429932, 2150.419922, 2160.399902, 2170.379883, 2180.360107, 2190.340088,
 2200.310059, 2210.280029, 2220.250000, 2230.219971, 2240.179932, 2250.139893,
 2260.100098, 2270.050049, 2280.000000, 2289.949951, 2299.899902, 2309.850098,
 2319.790039, 2329.729980, 2339.659912, 2349.600098, 2359.530029, 2369.459961,
 2379.389893, 2389.310059, 2399.229980, 2409.149902, 2419.070068, 2428.979980,
 2438.889893, 2448.800049, 2458.709961, 2468.610107, 2478.510010, 2488.409912,
 2498.310059, 2508.199951}
```

Figure 5.6. The content of the Header file, Lake Tahoe.hdr. It contains all the relevant information to be used for image loading.

b. *io package*

The io package is used to load image data according to the Header file information. In Figure 5.7, the left hand side of the window presents the Header file information and then right hand side of the window allows the users to choose from the bands available. The user can choose up to three bands to map into Red, Green and Blue colors. The listed numbers are the wavelengths for each band number.



Header Information

Description	AVIRIS null Tahoe
Samples	614.0
Lines	512.0
Bands	224.0
Header Offset	0.0
File Type	ENVI Standard
Data Type	2.0
Interleave	bsq
Sensor Type	Unknown
Byte Order	0.0
Default bands	8.0

Available Band List

```
2409.14990234375
2419.070068359375
2428.97998046875
2438.889892578125
2448.800048828125
2458.7099609375
2468.610107421875
2478.510009765625
2488.409912109375
2498.31005859375
2508.199951171875
2508.199951171875
```

Select bands

RED	383.1499938964844
GREEN	1361.93994140625
BLUE	2508.199951171875

Figure 5.7. Available band list view selection.

The image file data type is usually a 16 bit Signed Integer. The data ranges between –32768 and 32767. They can be in either IEEE or Intel standard. For some files used in Java application, byte swapping was necessary to convert from the Intel standard to the IEEE standard. Figure 5.8 shows the detail byte swapping operation. In this figure, two bytes are read from disk. The second byte of bigEndian is shifted into

8 bit left and the first byte is 8 bit right. Bit Or operation is applied after that. This swapping maintains the signed bit at the initial first byte as well.

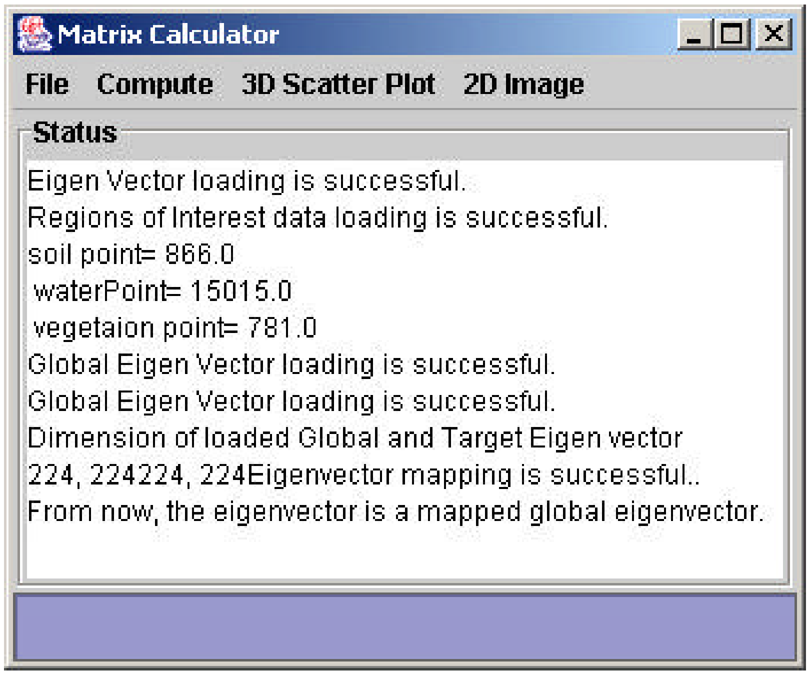
```
bigEndian = inDataStream.readShort();
         swapped = (int)   ( bigEndian<<8 ) | ( (bigEndian>>8) &
0xff);
         swapped = (swapped & 0x0000ffff);
 redMatrix[y][x] = swapped;
```

Figure 5.8. Byte swapping in ImageReader.java.

c. ComputationGroup package

This package provides all the required computation capabilities shown in Figure 5.1. A key operation is the matrix multiplication between Eigenvector and Region of Interest (ROI) data to produce the data to project, which is the PC data of ROI in Figure 5.1. This is the matrix multiplication between (m by n) of Eigenvector and (x by y) of either entire image data or ROI image matrix. This package uses the Java Matrix Package (JAMA) Application Program Interface (API), a open source for matrix operation (JAMA, 2001). For detail mathematics, refer to (Richard, 1995).

Of note is the data format and type when the resulting data are written back to disk. The data format is converted to Band Interleaved by Pixel (BIP). The data type is converted to Double, 32 bit floating point as the java primitive data type. The double type is used since it can be converted to string type. String data type are useful, since these data types can be inserted into the Document Object Model (DOM) tree. The detail conversion process is explained in the next section.



```
Eigen Vector loading is successful.
Regions of Interest data loading is successful.
soil point= 866.0
 waterPoint= 15015.0
 vegetaion point= 781.0
Global Eigen Vector loading is successful.
Global Eigen Vector loading is successful.
Dimension of loaded Global and Target Eigen vector
224, 224224, 224Eigenvector mapping is successful..
From now, the eigenvector is a mapped global eigenvector.
```

Figure 5.9. Snap shot of Matrix Calculator, the topmost class, in computationGroup package.

d. DomGroup Package

This is a key package for the 3D scene generation. It converts a base scene X3D file, DefaultHSVScatterPlot.x3d, into a DOM tree and inserts the manipulated data into this tree. This tree is then transformed into Virtual Reality Modeling Language (VRML) by the X3dToVrml97.xsl stylesheet. The original X3D and transformed VRML versions are each written back to disk. Either VRML-capable browsers or X3D-capable browsers can be invoked to load this X3D/VRML scene from disk.

e. Xj3DviewersGroup Package

This package provides one way to view the VRML scene created by the above DomGroup package. It contains a Xj3D Player, an open source to view a 3D scene. The Xj3D is mentioned in Chapter III. In addition to the Xj3D viewer, this package provides the other way to view 3D scene through a VRML capable Internet Explore (IE) with a Cortona plugin (Parallelgraphics, 2002). This alternative way was provided, since the Xj3D Player was not complete at the time the author developed this application (Xj3D, 2002).

f. *HistogramGroup Package*

This package is capable of viewing a 2D resultant image and writing it back to disk. It was named in the initial development phase to reflect the idea in the author's mind to provide the capability of viewing the histogram of image statistics with a 2D resultant image view as in Figure 5.10. The histogram plot was supposed to provide interactive stretching capability for Saturation and Value. However, this capability is currently mechanically performed and will be developed for future improvement later. The BufferedImage class is used with bit packing to Standard Red Green, Blue (SRGB), introduced in Chapter III. Figure 5.10 shows the result of Tyo's algorithm applied to the Lake Tahoe scene with 0 degree RGB rotation.

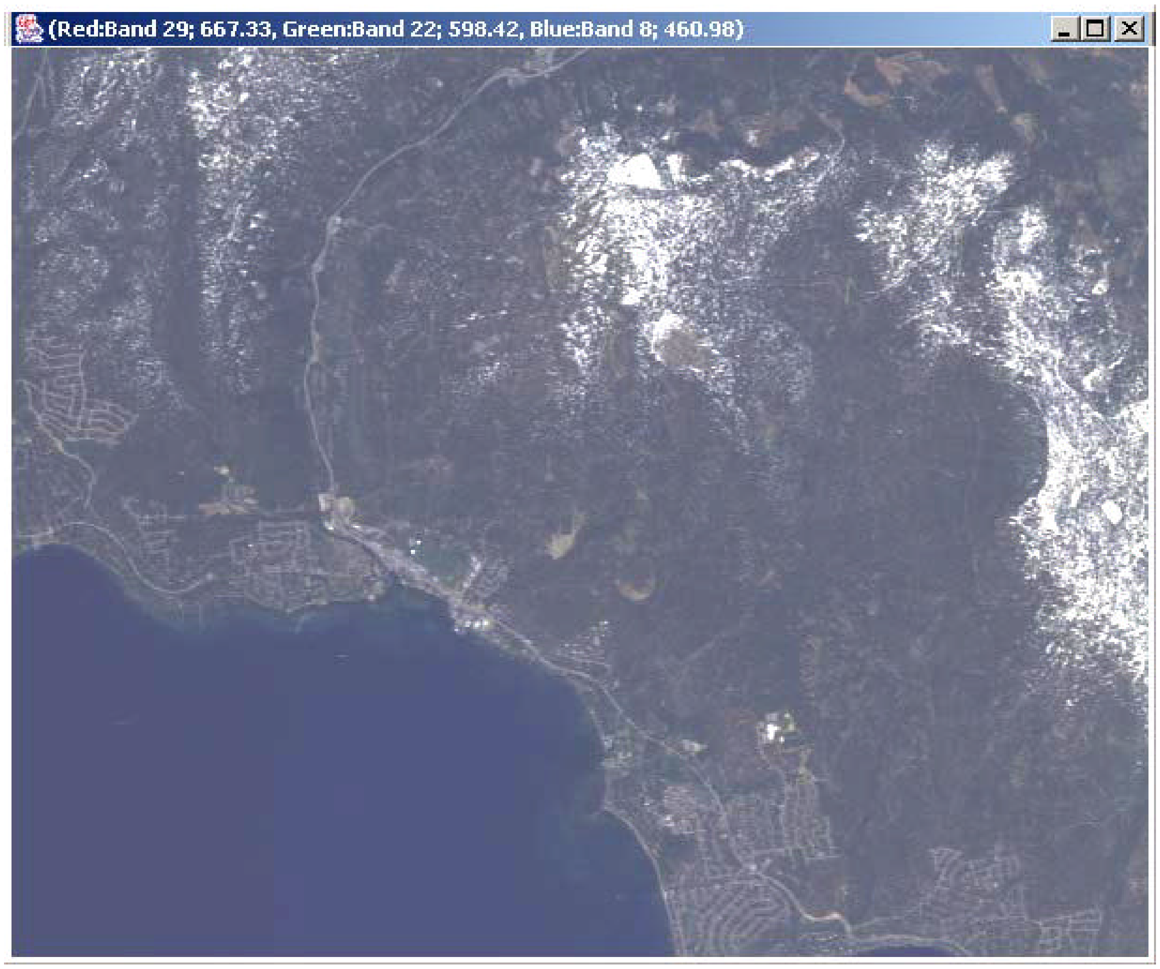

(Red:Band 29; 667.33, Green:Band 22; 598.42, Blue:Band 8; 460.98)

Figure 5.10. 2D image of 2D resultant image of Lake Tahoe after Tyo's algorithm was applied. The hue was not rotated at this scene.

D. 3D IMPLEMENTATION.

This section describes in detail how an X3D scene is generated. The first step for generating an X3D scene is to load an X3D file to memory in the Document Object Model (DOM) by using Java API for XML Parsing (JAXP). Then, the loaded DOM tree is modified using standard interfaces provided by JAXP. Once complete, the modified DOM tree is saved as an X3D file and then converted to VRML97 encoding. Next, the VRML97 scene is viewed by a Xj3D or VRML-capable browser. Figure 5.11 shows the entire procedure.

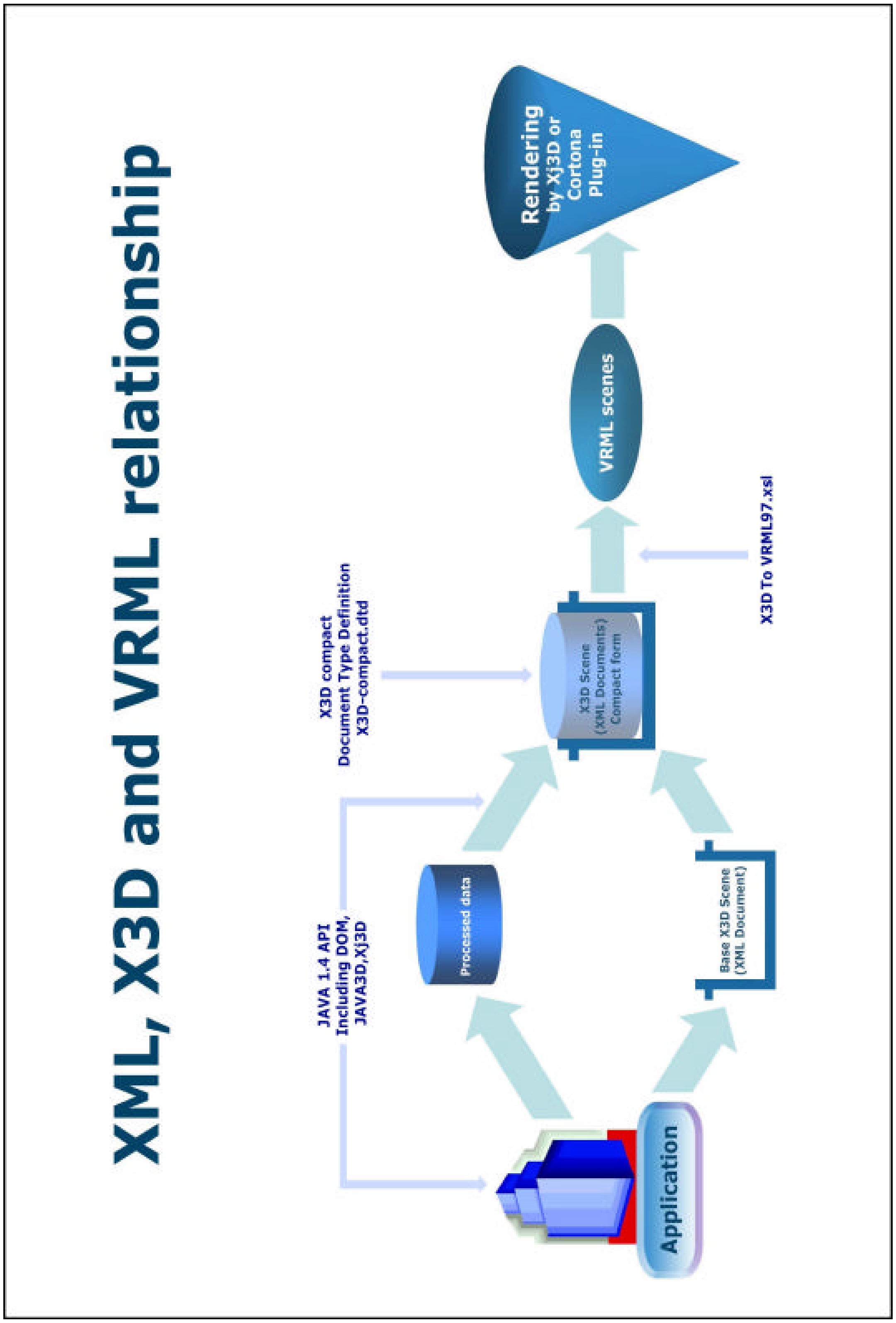


Figure 5.11. 3D scene generation process from Java to VRML through X3D.

1. Loading the Base Scene, DefaultHsvScatterPlot.x3d into Memory

DefaultHsvScatterPlot.x3d is a X3D file in XML format and was introduced in Chapter III. XML documents, when parsed, are represented as a hierarchical tree structure in memory as mentioned in Chapter III. This tree structure contains the document's elements, attributes and content. A programmer can add data, remove data, and query for data, since XML was designed to be dynamic. W3C provides a standard recommendation for building a tree structure in memory for XML documents called the Document Object Model (DOM). Each element and attribute in a XML document is represented by a node in the DOM tree and can be modified by manipulating the nodes in a DOM tree.

The DOM tree is created by Java API for XML Parsing (JAXP) provided by Sun Microsystems. JAXP uses the DocumentBuilderFactory class to create a DocumentBuilder object. The Class DocumentBuilder in Figure 5.12 provides a standard interface to an XML parser. The DocumentBuilderFactory produces an appropriate DocumentBuilder object for a currently configured XML parser. The builder object in Figure 5.12 provides an interface for loading and parsing an XML document. The method `parse()` is used to load and parse the XML document stored in the datafile, DefaultHsvScatterPlot.x3d, in Figure 5.3.

```
String filename = “DefaultHsvScatterPlot.x3d”;
File datafile   = new File(filename);
DocumentBuilder builder = factory.newDocumentBuilder();
document = builder.parse(datafile);

/***********************************************************************
* Build Document before manipulating point/color attribute in each node
***********************************************************************/
  Node root = document.getDocumentElement();
  Element x3dNode = (Element) root;
```

Figure 5.12. Code Snippet from VRMLMaker.java. These lines load an X3D file in XML format and put it in a Document Object Model (DOM) tree.

Upon the successful loading of the X3D file, a getDocumentElement method is called to obtain the Document's root node. The root is downcast to Element and the scene graph tree is traversed to find the Coordinate and Color node as shown in Figure 5.13 for further attribute value manipulation.

2. Modifying Point and Color Attribute Values

After downcasting the Node root to Element, methods specific to class Element can be called on the object using the x3dNode. The method, getElementsByTagName ("Coordinate") returns the list of all the Coordinate elements in the XML document. Each element is stored as an item in a NodeList. The first item added is stored at index 0. This index is used to access an individual item in the NodeList. Then, the StringBuffer pointValue is instantiated to append vegetation location values for x, y, and z values in the point attribute. Note that the current 2D array matrix has N by 3 dimension, where N is the number of all the vegetation points. After all the point values are appended into the StringBuffer, the pointValue is converted into a String to set the attribute of coordinate Element. The exact same mechanism is applied in the Color Element, except that the RGB values, according to the classification of the material, which is vegetation in this case, are set as attribute values. For instance, vegetation color Element attribute values are set as (0,1,0) for RGB to represent green, soil color is (1,0,0) as red, and water is (0,0,1) as blue.

```
//========================================================
//Modify the "POINT" attribute value for "COORDINATE" node
//========================================================
 NodeList coordinateNode = x3dNode.getElementsByTagName("Coordinate");
 Element coordinate = (Element) coordinateNode.item(0);
 StringBuffer pointValue = new StringBuffer();

  for (int i = 0; i < hsvGenerator.vegetation.length; i++)
  {
     for (int j = 0; j < hsvGenerator.vegetation[0].length; j++)
     {
       pointValue = pointValue.append((int)hsvGenerator.vegetation[i][j]+" ");
     }
   }
 String point = (String) pointValue.toString();
 coordinate.setAttribute("point",point);
 //======================================================
 //Modify "COLOR" attribute value for the "COLORNODE"
 //=====================================================
 NodeList colorNode = x3dNode.getElementsByTagName("ColorNode");
 Element color = (Element) colorNode.item(0);

 StringBuffer colorValue = new StringBuffer();
 for (int i = 0; i < hsvGenerator.vegetationColor.length; i++)
 {
     for (int j = 0; j < hsvGenerator.vegetationColor[0].length; j++)
     {
        colorValue = colorValue.append(hsvGenerator.vegetationColor[i][j]+" ");
     }
 }  String colorString = (String) colorValue.toString();
 color.setAttribute("color", colorString);
```

Figure 5.13. Modification of point and color attribute values in VRMLMaker.java.

3. Transforming DOM to VRML97 Encoding

The Extensible Stylesheet Language (XSL) is used to format XML documents and consists of two parts: the XSL Transformation (XSLT) language and XSL formatting objects. XSLT is used to transform the X3D document into a VRML97 encoding. In this thesis, X3DtoVrml97.xsl was used as the conversion XSLT stylesheet. An XSLT document is an XML document with a root element stylesheet. The namespace for an XSLT document is http://www.w3.org/1999/XSL/Transform. The XSLT document shown in Figure 5.13 transforms DefaultHsvScatterPlot.x3d (Figure 5.3) into a VRML97 document (Figure 5.14).

```
<!-- ****** root:  start of file ****** -->
<xsl:template match="/">
  <!-- VRML 97 header -->
  <xsl:text>#VRML V2.0 utf8&#10;# X3D-to-VRML-97 XSL translation autogenerated
by X3dToVrml97.xsl&#10;#
http://www.web3D.org/TaskGroups/x3d/translation/X3dToVrml97.xsl&#10;&#10;</xsl:
text>
  <!-- VRML 200x headers -->
  <xsl:apply-templates select="X3D"/>
  <xsl:apply-templates select="X3D/head/component"/>
  <xsl:if test="X3D/head">
    <xsl:text>&#10;</xsl:text>
    <xsl:apply-templates select="X3D/head"/>
  </xsl:if>
  <xsl:if test="X3D/Header">
    <xsl:call-template name="output-error">
      <xsl:with-param name="errorString">
        <xsl:text>'Header' tag illegal, use 'head' instead</xsl:text>
      </xsl:with-param>
      <xsl:with-param name="node">
        <xsl:text>X3D/Header</xsl:text>
      </xsl:with-param>
    </xsl:call-template>
  </xsl:if>
<!-----etc ----→
```

Figure 5.13. Code snippet of X3dToVrml97.xsl used in this thesis.

```
#VRML V2.0 utf8
# X3D-to-VRML-97 XSL translation autogenerated by X3dToVrml97.xsl
# http://www.web3D.org/TaskGroups/x3d/translation/X3dToVrml97.xsl

# [X3D] VRML V3.0 utf8

# [head]
# [meta] filename: Default.x3d
# [meta] author: Kang Kim
# [meta] revised: 20 Oct 2002
# [meta] description: A scatter plot with the points colored using colors from a Color
node.
# [meta] generator: X3D-Edit,
http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html
# [Scene]

# Positive direction of X axis in X3D space is the base axis. Counter Clock Wise(CCW)
rotation increases angle. Clock Wise(CW) decreases angle.
Viewpoint {
  description "Front view points"
  jump FALSE
  position 5 4 80
}
Viewpoint {
  description "Top view point"
  jump FALSE
  orientation -1 0 0 1.57
  position 30 80 30
}

NavigationInfo {
  type [ "EXAMINE" ]
}
Group {
  children [
      Shape {
        geometry PointSet {
          coord DEF originalPoints Coordinate {
            point [ -3181.22  588.94 -197.67 ]
          }
          color Color {
            color [ 1  0  0 ]
          }
        }
      }
      Transform {
        scale 20 20 20
        children [
            Inline {
              url [ "PC_CoordinateAxes.wrl" ]
            }
        ]
      }
  ]
}
```

Figure 5.14. Conversion result of DefaultHsvScatterPlot.wrl.

To process XSLT documents, an XSLT processor is required. The open source Apache Foundation has created the Xalan XSLT processor for Java. (Apache XML Project, 2002) Xalan-Java is an XSLT processor for transforming XML documents into other XML document types. It implements the W3C recommendations for XSL Transformations (XSLT) and the XML Path Language (XPath). An open-source product, use of this toll is available free on all programming platforms.

4. Viewing a Model in an X3D/VRML Capable-Browser

This section shows how a VRML scene was loaded into a Xj3D Browser. A standard loader as defined by Sun's utility interface `com.sun.j3d.loaders.Loader`, (Xj3D Task Group, 2002), and was used as a loading method. Implementing a loader means that the code is required to obey a set of flags, and also to do most of the work within the loader, rather than by application code.

When loading external files, the simpler memory-based `MemCacheLoadManager` is used for processing external files since the environment in which the loader is used may be unknown. The loader implementation automatically determines the file type to be loaded. It handles VRML97, X3D and VRML3.0 for users.

Constructing a new loader requires creating an instance of the class `org.web3d.j3d.loaders.VRMLLoader`. There are two standard constructors available: the default no-argument constructor and one that takes an `int` argument, which are the load flags. The example below used int Xj3DBrowser.java to create a loader that builds a fully compliant VRML scene graph:

```
import java.io.IOException;
import com.sun.j3d.loaders.Loader;
import com.sun.j3d.loaders.Scene;
import org.web3d.vrml.j3d.VRMLLoader;
....
  Loader ldr = new VRMLLoader(Loader.LOAD_ALL);
```

Once a loader is constructed, the next step is to load one or more files. This is done through the usual assortment of `load()` methods. Construct a URL or file path and pass it to the load method. In return, the code renders a `Scene`.

```
try
{
  Scene vrml_scene = ldr.load
         ("d:/JavaProjects/ImageXj3D/SoilVegetationWater.wrl");
}
catch(Exception e)
{
  //Exception handling
}
```

To access the loaded Java3D scene, the `getSceneGroup()` method is then called. The resulting `BranchGroup` can then be placed in an application's scene graph.

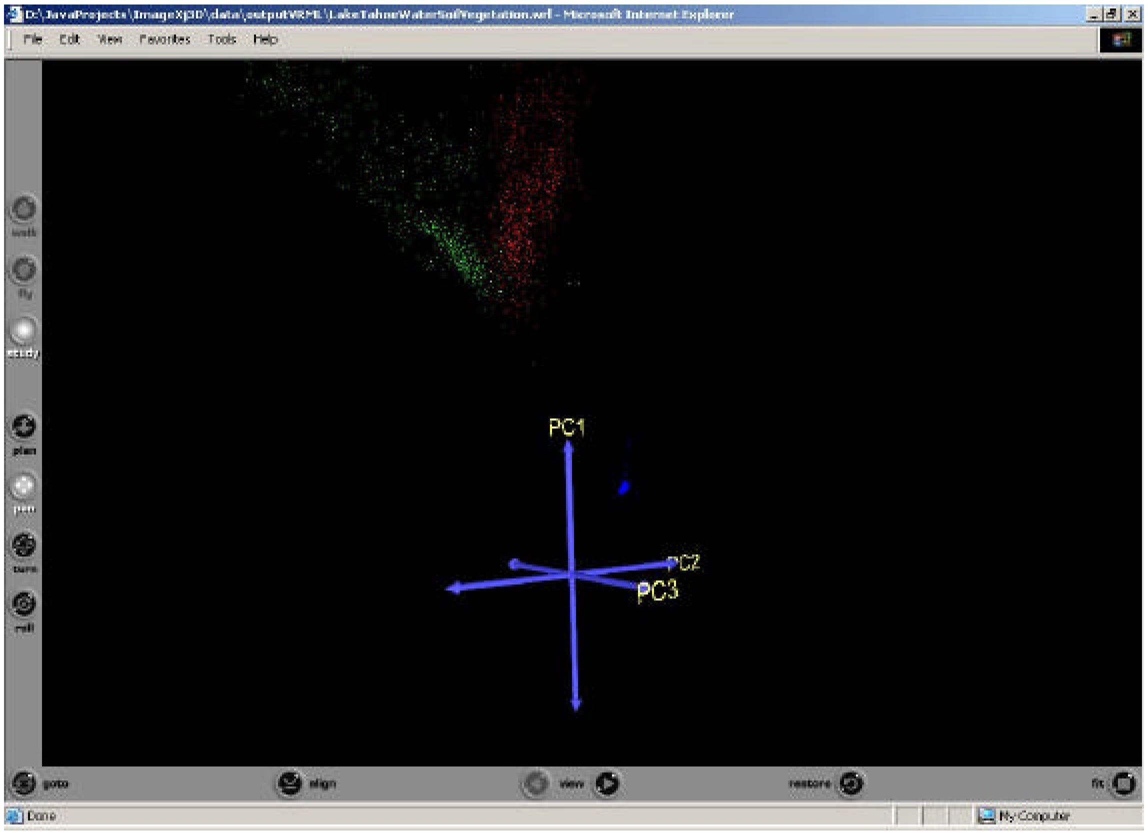

D:\JavaProjects\ImageXj3D\data\outputVRML\LakeTahoeWaterSoilVegetation.wrl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
walk
fly
study
plan
pan
turn
roll
PC1
PC2
PC3
goto
align
view
restore
fit
Done
My Computer

Figure 5.15. Scatterplots of generated 1st, 2nd, 3rd PC of soil, water, vegetation in PC data space. A. Cone Projected onto the P_2 - P_3 Plane. B. P_2 – P_1 Plane. C. P_3 - P_1 Plane. The numerical value presented here represent absolute projections of the HSI data at each pixel onto the eigenvectors of the covariance matrix of the data with no scaling or translation. (From: Tyo et. al)

E. SUMMARY

This chapter introduced the overall application structure. The great detail of the 3D construction and viewing methods used in this thesis were presented. The final 3D scene constructed in Figure 5.15 will be analyzed in the follow chapter.

VI. DATA ANALYSIS

A. INTRODUCTION

This chapter analyzes the resulting 3D conical data shape that represents the dataset to HSV mapping. The implementation result helps to verify the suggested strategy by developing In-Scene Global Statistics and applying a post-rotation of RGB encoded PC transforms. The case study method is used in this chapter, showing different images acquired from different sensors including the Hyperspectral Digital Imagery Collection Experiment (HYDICE), the Airborne Visible/Infrared Imaging System (AVIRIS), the Night Vision Imaging System (NVIS) and the Hyperspectral Mapping system (HyMap).

B. 3D VISUALIZATION ANALYSIS FOR HSV CONICAL REPRESENTATION OF PRINCIPAL COMPONENT (PC) DATA

The conical shape of PC data is used to represent ROI Hue, Saturation, and Value (HSV) color space. Figure 6.1 shows the hexagonal HSV color space and ROI data. ROI data also was projected as red, green, and blue scattered points. These points were computed by PC Transformation of the Lake Tahoe scene Eigenvector on soil, water, and vegetation points on the same scene. Visually comparing the green axes in the Hue plane with the mapped green points revealed that there exists about a 30 degree difference between the green color vertex in HSV space and the location of vegetation points calculated from Tyo's algorithm. In other words, the vertex of green color in the HSV color space is located at a Hue angle of 120 degree. The location of green points from Tyo's algorithm appears to be at 150 degrees. Thus, a -30-degree rotation in Hue plane is required for a more intuitive mapping of the eigenvector data space. This additional rotation requirement is applied to the RGB transformation in the following section.

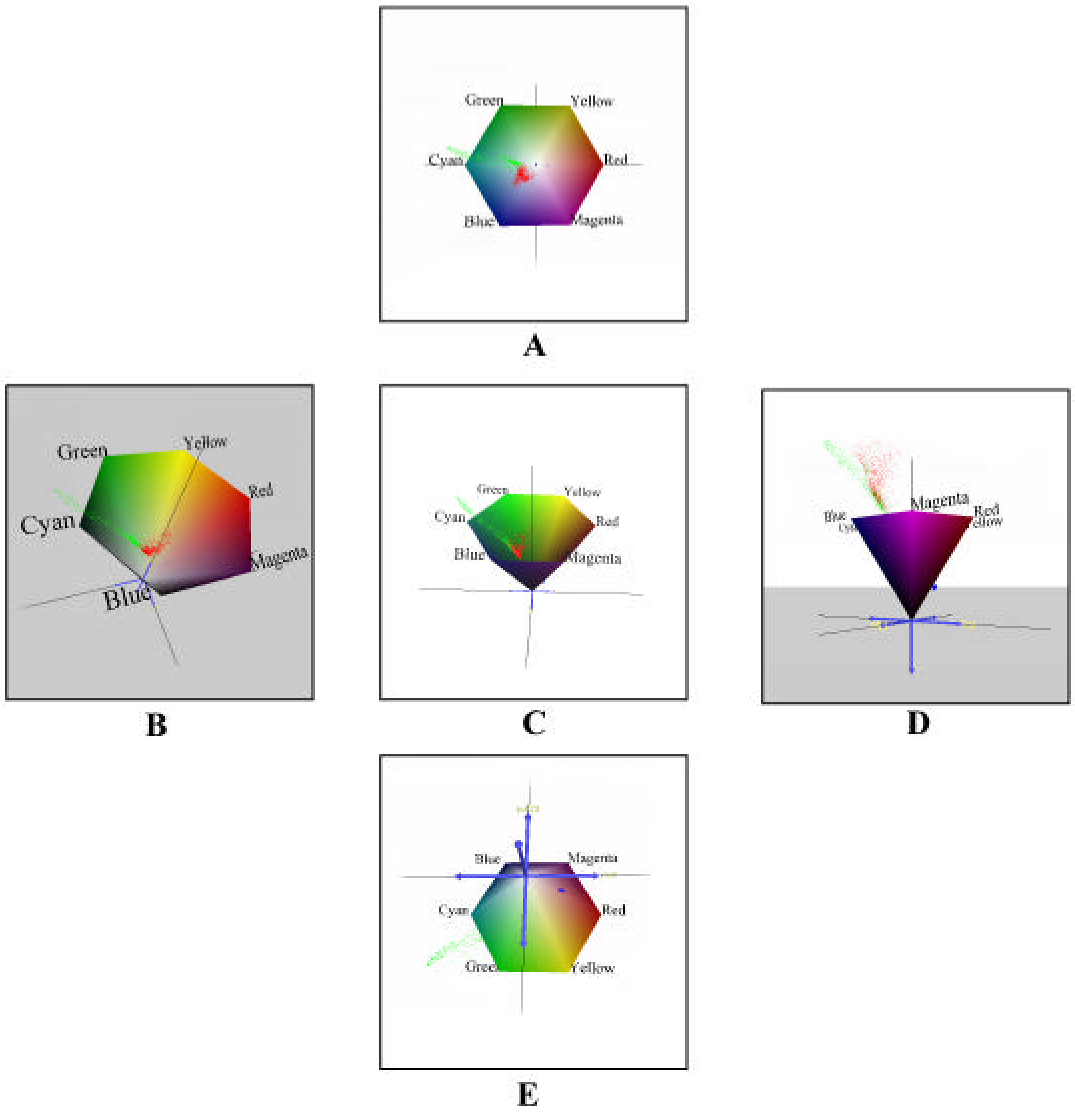


Figure 6.1. ROI projection in HSV color space. A. Top view, B. View tilted 45 degree to left. C. Front view, D. View tilted 45 degree to right, E. Bottom view.

C. DEVELOPING GLOBAL STATISTICS (EIGENVECTORS)

The first section identified the required additional Hue angle rotation. Before applying this rotation angle on the RGB transformation, it is necessary to explain how In-scene statistics are used and how they are developed.

In this thesis, the Lake Tahoe scene is used to produce the In-Scene statistics. Lake Tahoe was selected since it has relatively simple scene constituents such as vegetation and water. Figure 6.2 shows Hue rotated images, natural color image, typical false color in the Infrared image, natural color images and the true color image of the Lake Tahoe scene. The following sections explain the differences between each eigenvector.

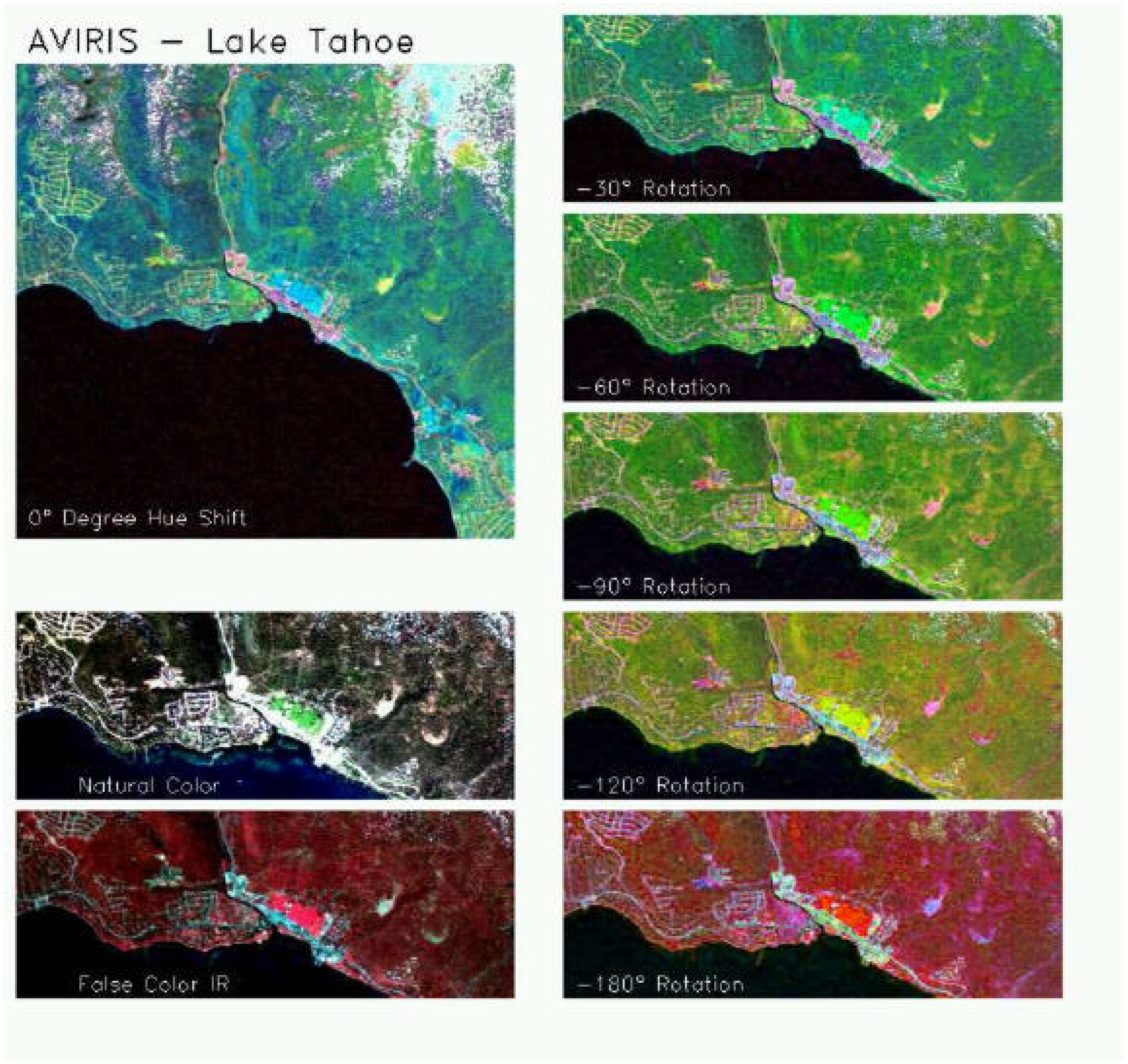


Figure 6.2. Overview of Lake Tahoe scene manipulation. Clock wise fom the top left, the image shows Post-Rotation of RGB Transformation between 0 to –180 degree, False Color in Infrared, and Natural color.

1. Scene Statistics and Eigenvector

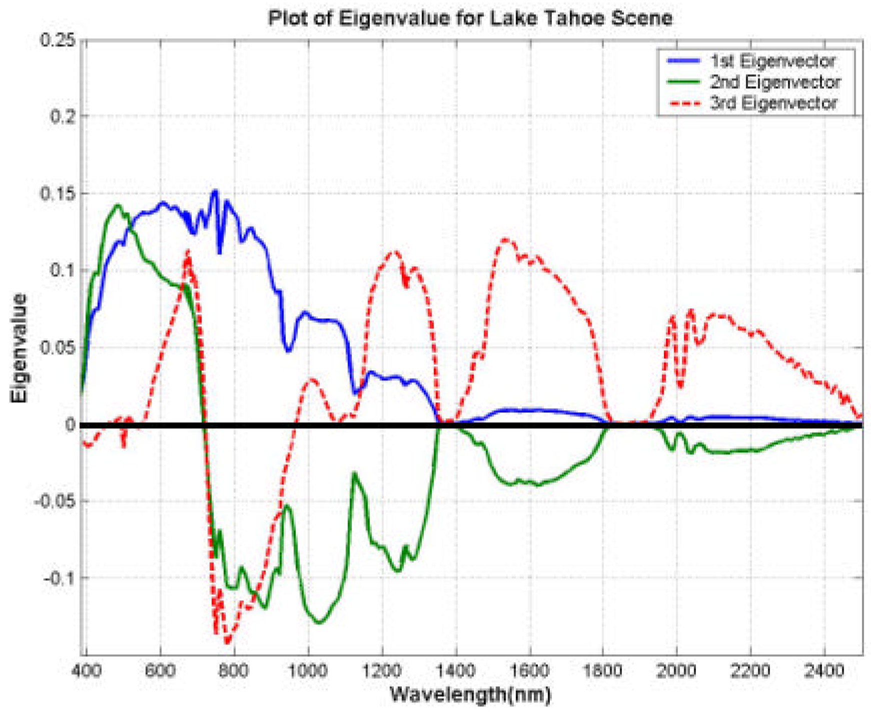
The Eigenvector derived from this AVIRIS scene was defined to be the 'global' eigenvector to be applied to successive data sets. The invariant display strategy pursued here depends on the structural similarity of the first few eigenvectors. The first three

Eigenvectors of each scene tend to behave similarly since they correspond to the statistical correlation between the bands, not that of individual scene signals. Figure 6.3 illustrates this behavior. Figure 6.3.A. shows the first three eigenvectors of the Lake Tahoe scene and Figure 6.3.B. shows the eigenvector for Davis Monthan Air Force base. As shown in Figure 6.3, these two scenes consist of different scene constituents. The Lake Tahoe scene is dominated by water and vegetation, while Davis Monthan Air Force base is sand and part grass. Note that the overall sign of the eigenvector is not important – the second eigenvectors in Figure 6.3A and Figure 6.3B are otherwise similar in shape.

In previous work, the average of sixteen different scenes was taken to compute a Global Eigenvector. On the contrary, in this thesis, only the Eigenvector of the Lake Tahoe scene is taken and this Eigenvector was mapped to other scenes acquired from different sensors instead, as mentioned in the Introduction. This thesis takes a different approach by applying these global statistics to the different sensor's scenes. The next section shows the results of applying these global statistics to other image data and of the post-rotation of RGB transformation.

2. Mapping Eigenvectors

Direct application of the Lake Tahoe eigenvector to other scenes is not always feasible. Mapping a chosen global statistic to individual scene eigenvectors is typically required, since the number of bands are different for different sensors. For instance, the HYDICE collects 224 bands and AVIRIS collects 210 bands. In this thesis, the closest value of the global statistics for the target scene Eigenvector is mapped into the respective band.



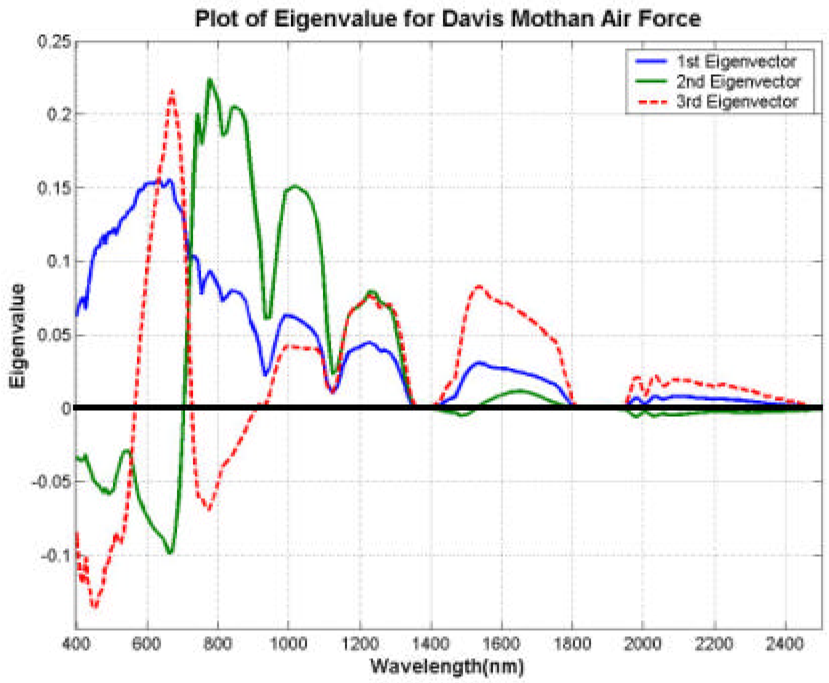


Figure 6.3. A: First three Eigenvectors plotted for the Lake Tahoe Scene. B. First three Eigenvectors plotted for the Davis Monthan Scene.

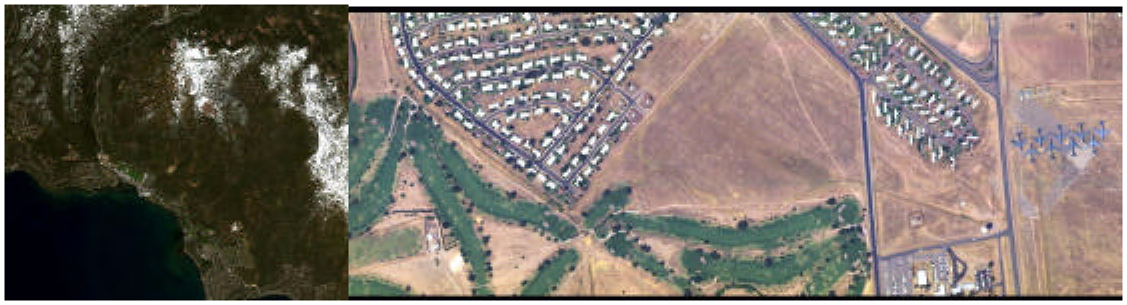


Figure 6.4. A. Lake Tahoe, B. Davis Monthan Air Force base.

D. CASE STUDY FOR POST ROTATION OF RGB TRANSFORMATION

This section presents the result of post rotation of RGB Transform, to verify the 30-degree Hue angle rotation acquired from the second section. Prior to RGB rotation, the global eigenvector acquired from the Lake Tahoe scene is mapped into other scenes according to the previous section. HSV data is computed based upon this PC data through Tyo's algorithm.

As mentioned in the introduction of this chapter, different types of images are used to test a global statistics and post-rotation of RGB transform. These are images acquired by NVIS, AVIRIS, HYDICE and HyMap sensors. Each sensor collects the different number of band information and different band ranges as well.

1. Davis-Monthan Air Force Base (HYDICE Data, 210 Bands)

Figure 6.5 shows the –30-degree rotation of RGB Transformation. Grass at the golf course appeared green as in real life. Unfortunately, the desert soil appears blue.

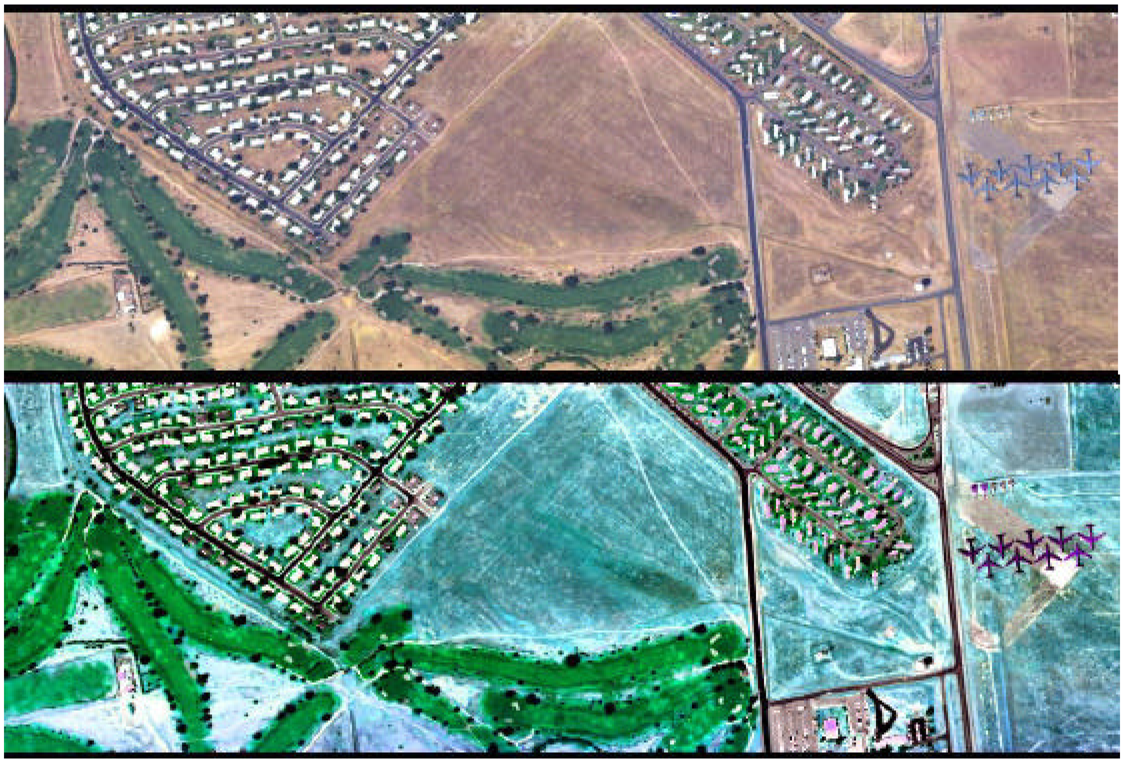


Figure 6.5. Davis-Monthan Air Force Base: A. Natural color, B. –30-degree rotation of RGB Transformation via HSV colorspace operations.

2. Jasper Ridge, California (1999, Hyperspectral Mapping (HyMap) Data, 60 Bands, Courtesy of Analytic Imaging and Geophysics (AIG) and HyVista Corporation)

For the HyMap scene, the vegetation again appears as green, the soil as light blue. The roads are red.

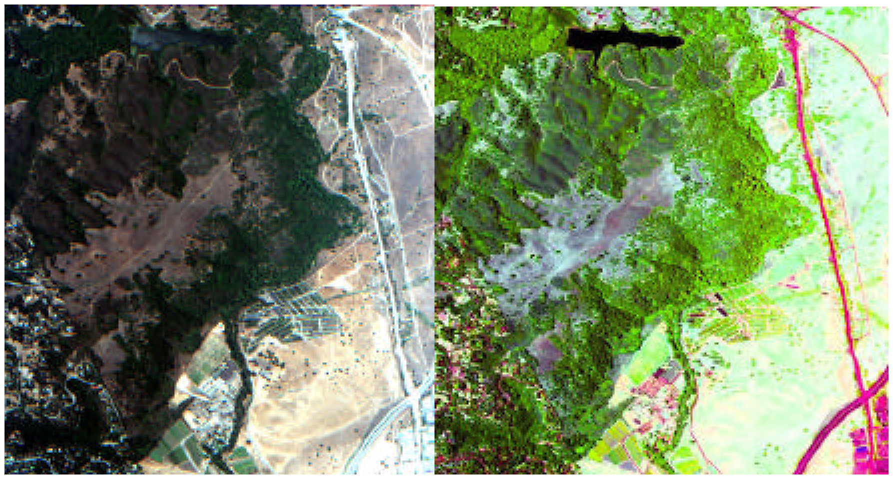


Figure 6.6. Jasper Ridge, California. A. Natural color image, B. –30 degree rotation of RGB Transformation of 1999 HyMap data of Jasper Ridge, California.

3. Camp Pendleton (HYDICE Data, 210 Bands)

For the HYDICE scene, the vegetation is again green emphasized by the two ellipses. The dry hillsides are blue. The damp sand shows a variety of colors.

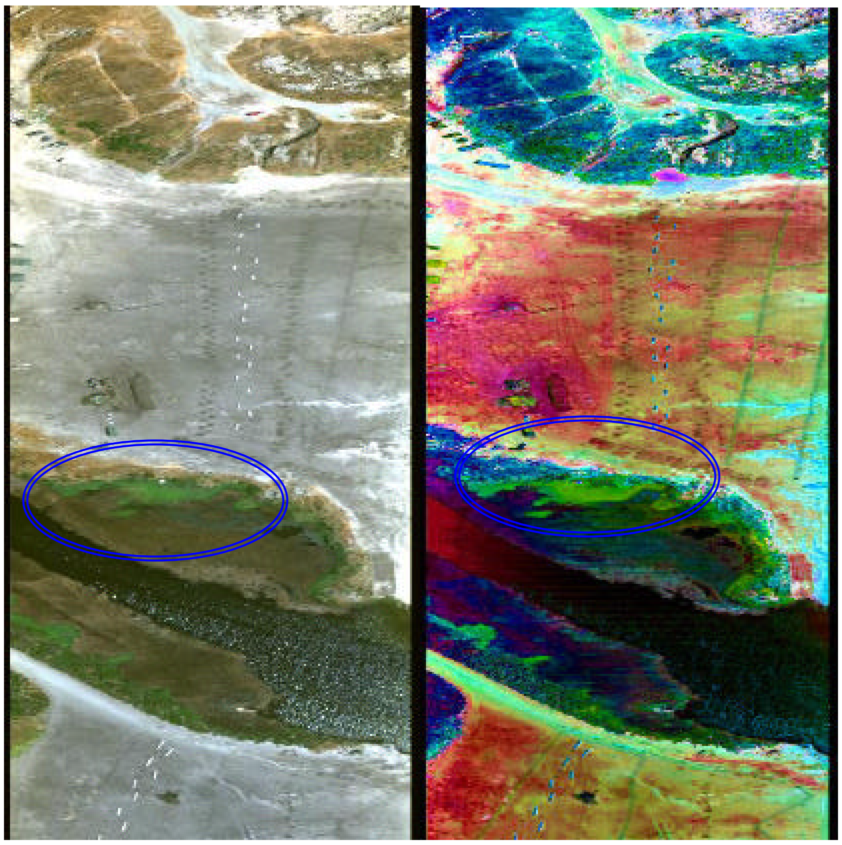


Figure 6.7. Camp Pendleton, California. A. Natural color image, B. –30-degree rotation of RGB Transformation.

4. Moffett Field, California (AVIRIS Data, 224 Bands)

The relatively urban scene shows a new range of colors, with green again defining vegetation.

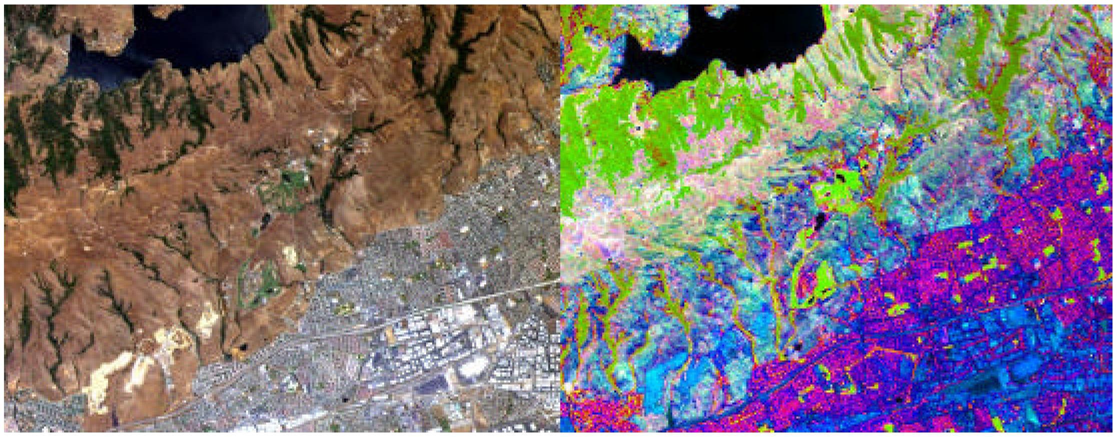


Figure 6.8. Moffett Field, California. A.: Natural color image, B. –30-degree rotation of RGB transformation.

5. Cuprite, Nevada (AVIRIS Data, 224 Bands)

The classic Cuprite scene has no vegetation. The different minerals are not qualitatively distinguished in this view. Such scenes may need an alternate transform.

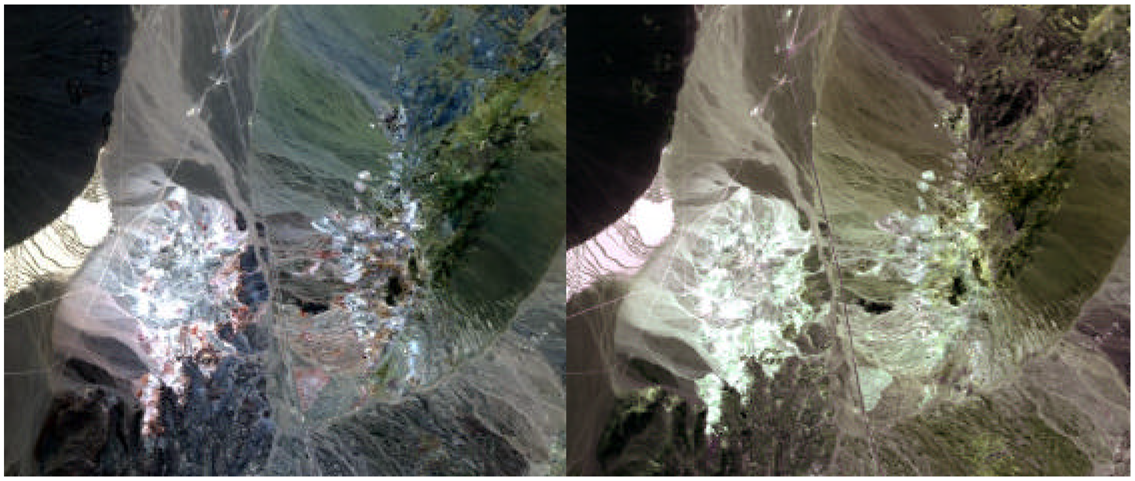


Figure 6.9. Cuprite, Nevada from the ENVI tutorial data set. A. Natural color image. B. –30-degree rotation of RGB Transformation.

E. DISCUSSION

In this section, one factor to provide for better observation of post rotation of RGB transform and non-compliant results of Tyo's algorithm are discussed.

1. Saturation and Value Contrast Stretch

The one factor being considered to provide for better observation is the HSV data contrast stretch. The hue variation is not reflected in the updated change if distribution of saturation and value is too broad. Figure 6.10 shows an example from Night Vision Imaging System (NVIS). The top of the trees in Figure 6.10 appear white in panel A and do not reflect the change of hue by rotation of the RGB transform in panel B.

To reflect the hue rotation changes, saturation was adjusted in HSV space before RGB conversion was done as shown in Figure 6.11. Panel A in Figure 6.11 shows the saturation distribution before modification by contrast stretch. The saturation image appears very dark, indicating that the saturation distribution is too low. To enhance this saturation behavior, the distribution ranges are stretched by setting the minimum at 0.00 and the maximum at 0.232 as shown in Figure 6.11 panel B. The results of adjusting for a fuller saturation are shown in figure 6.12. Figure 6.12 and Figure 6.13 show the range of colors derived by varying the rotation angle from +180 to –180.

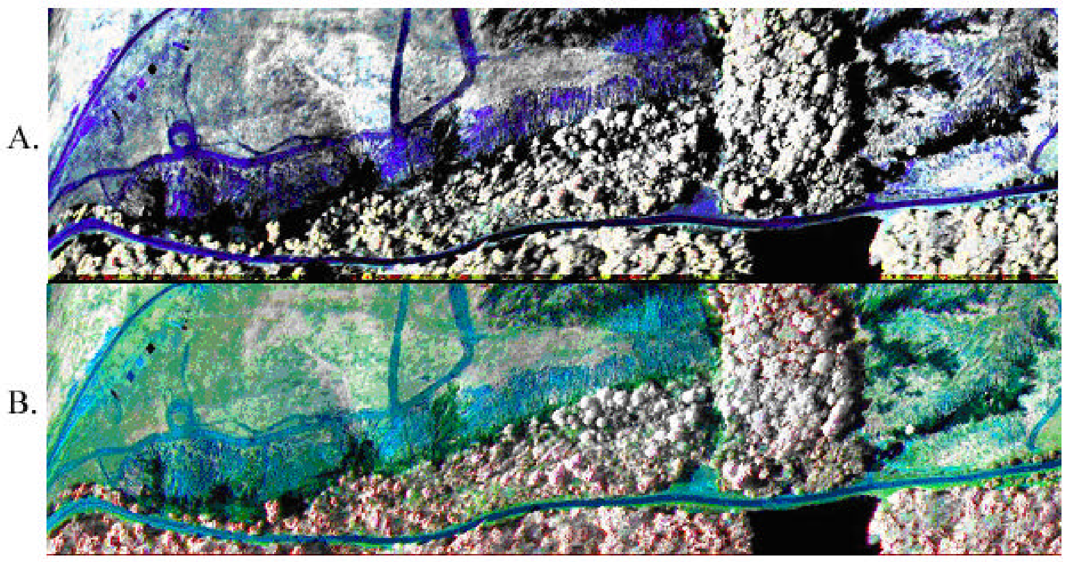


Figure 6.10. The desaturated examples without contrast stretch adjustment. A. Rotated 60 degrees after RGB transformation. B. Rotated –30 degrees after RGB transformation.

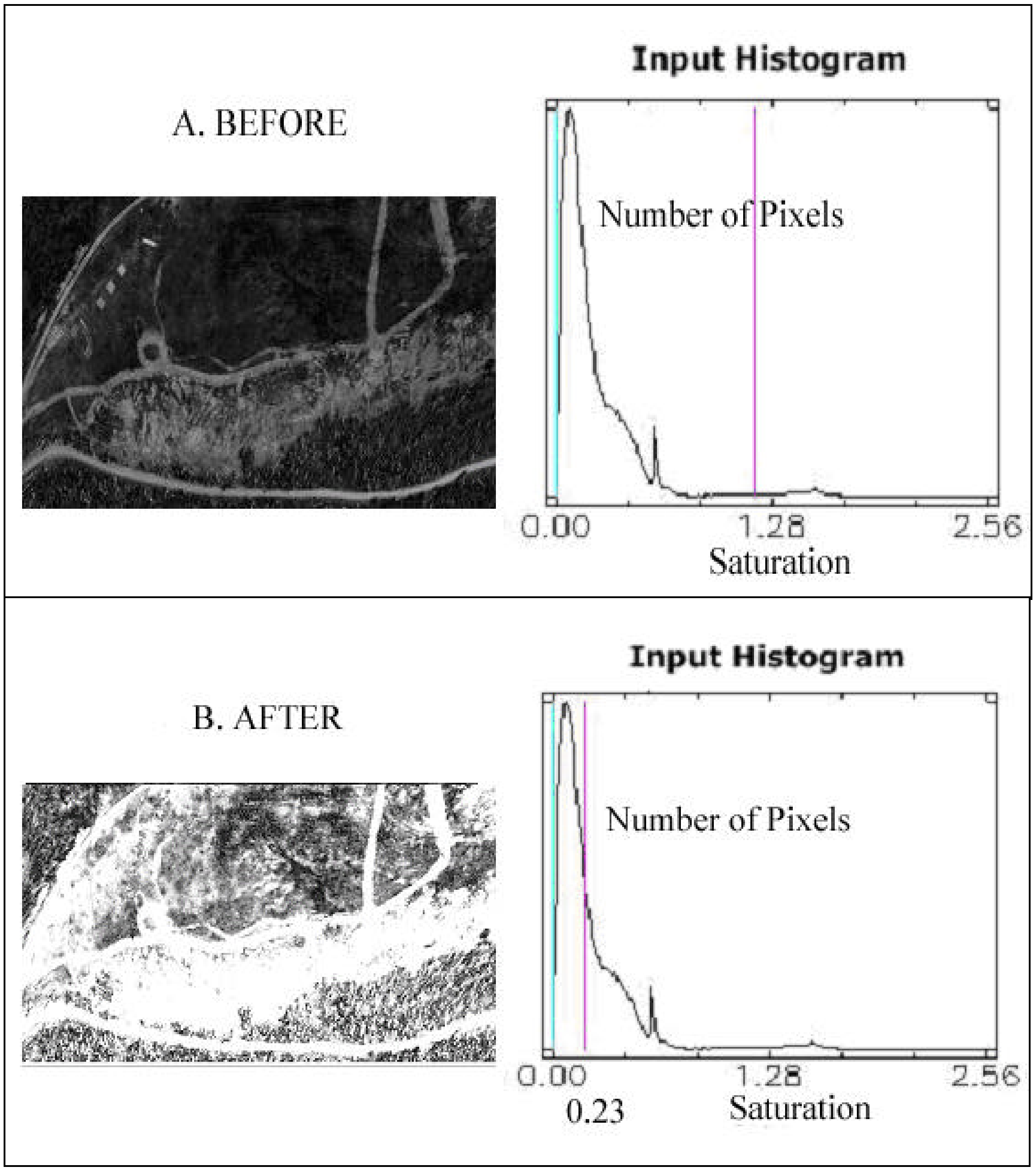


Figure 6.11. The enhancement of saturation through contrast stretch of histogram. A. Stretching Saturation value from 0.00 to 1.183. B. Stretching Saturation value from 0.00 to 0.23

Table 6.1 provides a summary of the minimum and maximum of saturation and the value for each data used in the previous section.

Table 6.1. Summary of Saturation and Value Ranges.

Image Location	Saturation		Value	
	Min	Max	Min	Max
Davis Monthan AF Base	0.067	0.39	31514.32	75922.12
Jasper Ridge, CA	0.078	0.257	1751.628	9345.87
Camp Pendelton	0.001	0.132	9748.2	60654.4
Moffett Field, CA	0.006	0.115	6769.4	39571.66
Cuprite, Nevada	0.125	0.162	1360.474	2322.716

2. Non Compliant Result to the Strategy

Two observations are made in this section. One is that the green color post rotation of RGB transform left the other regions in non-intuitive color. The second is that even a single material such as vegetation showed as different colors.

First, note the soil color in Figure 6.5 panel B. The soil between the golf course and the housing area all turned into blue. This is worth noticing, since the same effect appears in the remaining scene data sets, as well. The reason for this effect can be inferred from the first section, since the soil data was projected close to the blue vertex of the HSV color space.

Secondly, the reader might ask why the top of the tree and grass appears in a different color, even though they have the same chlorophyll material in Figure 6.12 and Figure 6.13. For instance, the tree and grass in panel B of Figure 6.13 appear purple and green respectively. This difference in color clearly indicates that different hue values exist between the grass and the top of the trees. This difference in spectra is shown in Figure 6.14. Several possibilities as to why they appear in a different color can be considered. Among them, the author considers one possibility that the distance between the grass and soil is different from that between the top of the trees and the soil. In other words, the soil material must affect the color of the grass, since grass does not completely cover the soil. Soil material radiance should be reflected in the color of the grass. Another possibility could be one that the grass might not be healthy as the tree. In addition, chlorophyll in grass could be producing different radiation values from those the tree is producing. However, these possibilities could be further investigated in future works.

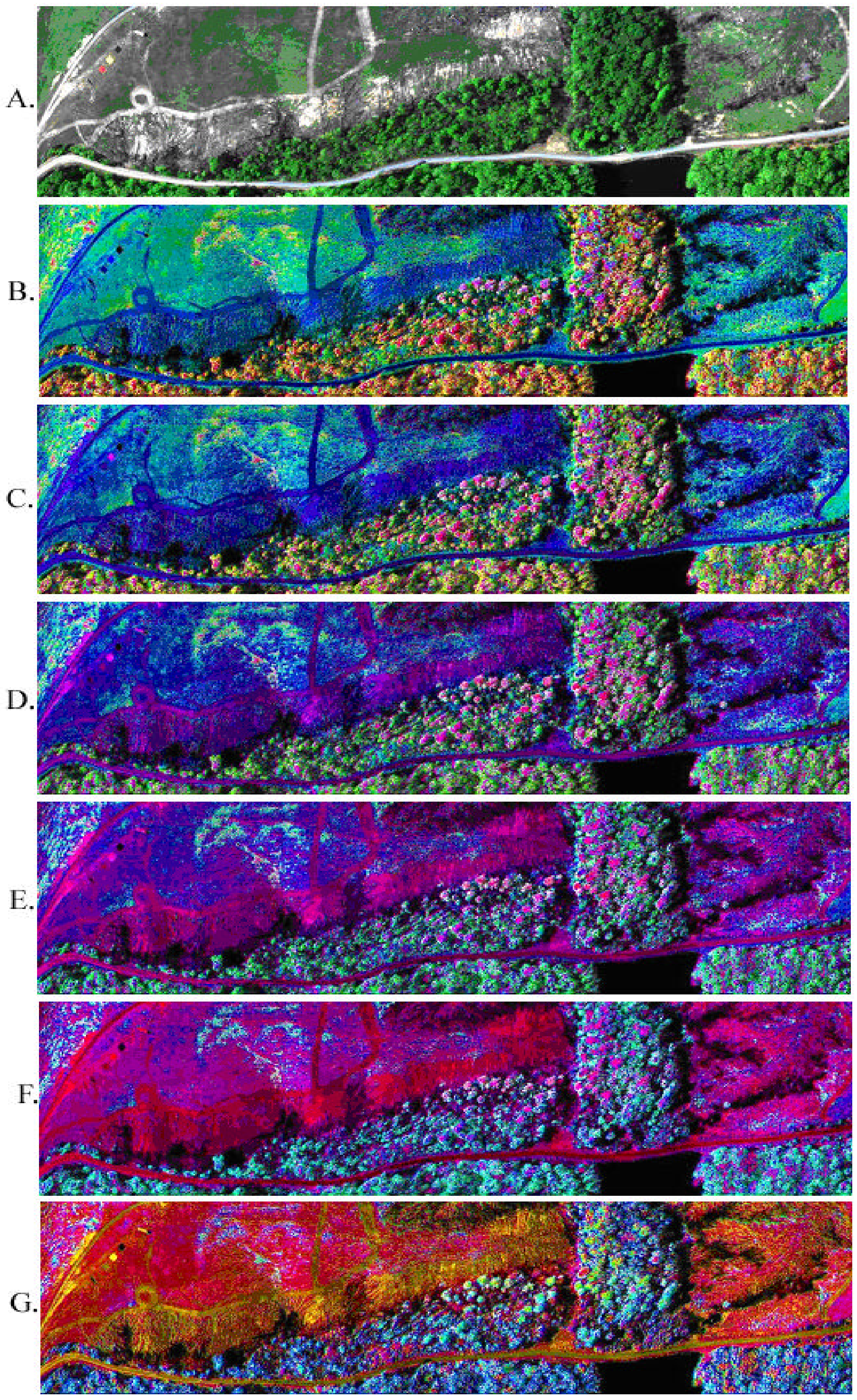


Figure 6.12. Post rotation of RGB transformation of NVIS image data to positive direction in Hue plane. A. 0 degrees. B. 30 degrees. C. 60 degrees. D. 90 degrees. E. 120 degrees. F. 150 degrees. G. 180 degrees.

The figure below illustrates the Hue rotation effect from –30 degrees toward -180 degrees.

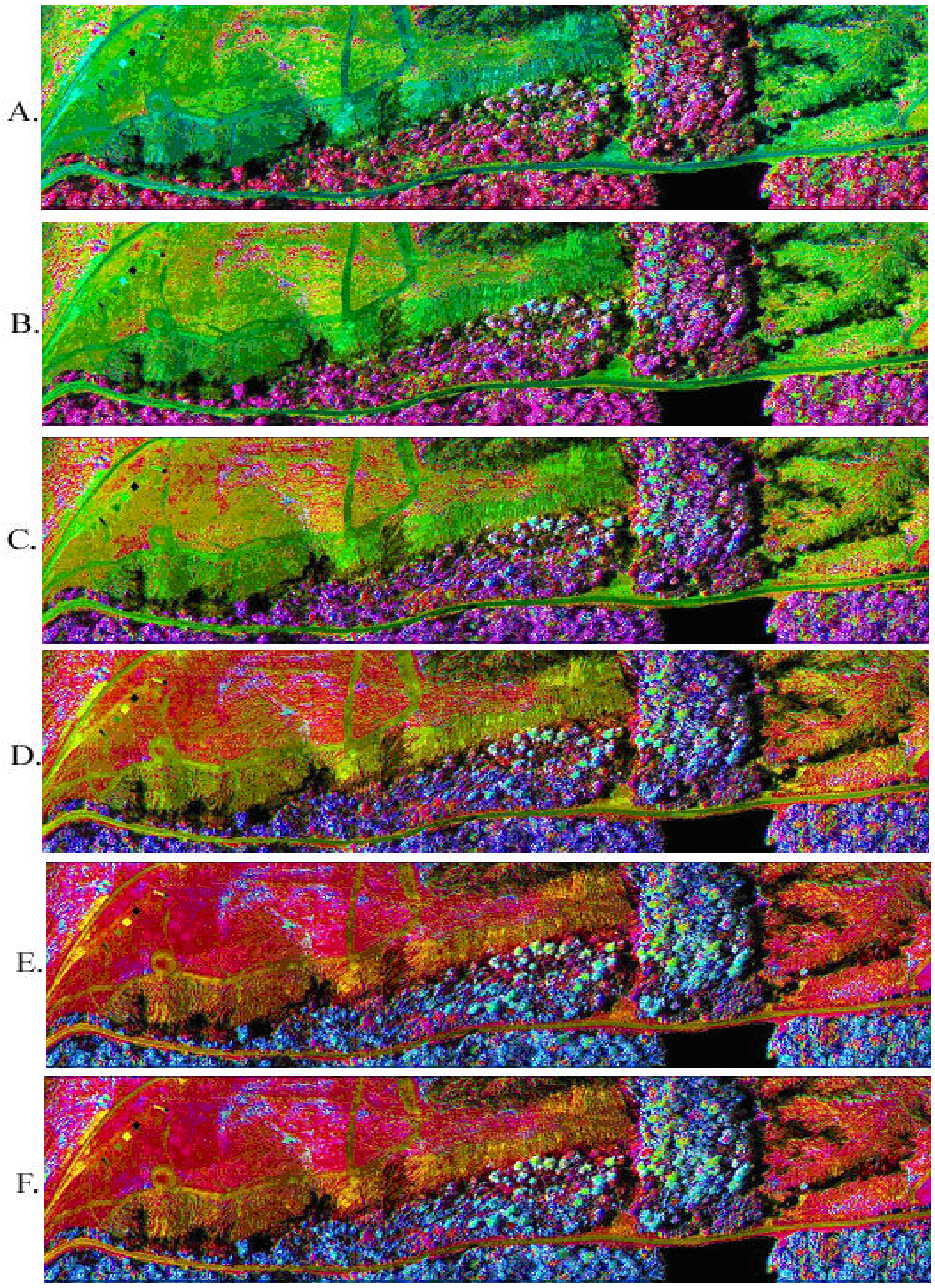


Figure 6.13. Post rotation to negative direction in Hue plane. A. –30 degrees. B. –60 degrees. C. –90 degrees. D. –120 degrees. E. –150 degrees. F. –180 degrees.

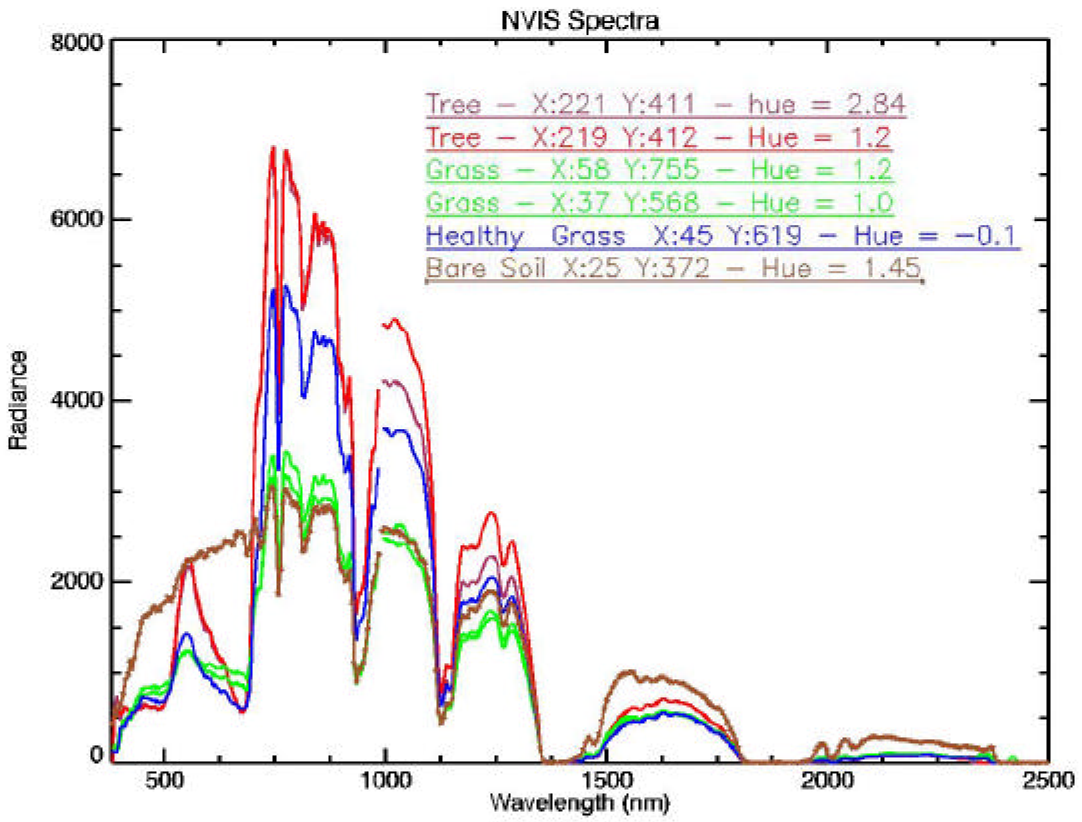


Figure 6.14. The spectral plot of the above NVIS data.

The above non-compliant results suggest the necessity to enhance the current application of Tyo's mapping algorithm. The current algorithm utilizes linear mapping when converting PC data to HSV. Thus, using non-linear mapping can be considered to eliminate the non-compliant behavior of the color of trees as shown throughout this section. In particular, non-linear stretches and rotation in hue are appropriate to study.

F. SUMMARY

The first section of this chapter showed the conical shape of PC data space in 3D. This projection of PC data identified required additional Hue rotation for the classification of vegetation. This angle had been applied to RGB transformed data to verify the effect. The effect of Hue rotation was nicely reflected. However, the non-compliant result of other colors brought up the issue of further enhancement of Toy's algorithm. Possible future work is implied, such as improving the mapping strategy from linear to a higher order.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

A. SUMMARY

Analysis of spectral component transform showed that the first three components could be considered in a new coordinate space defined in part by a conversion to Hue, Saturation and Value. This coordinates space can be thought of as a conical coordinate system, with the first Principal Component (PC), or intensity always along the z-axis. The color information (Hue and Saturation) are displayed as angle, and radius respectively. Here Hue and Saturation are taken from the second PC and third PC. Subsequently, the image data from this presentation was displayed in an RGB format from the HSV encoded principal component bands between one and three. A rotation in Hue was applied to cause the vegetation to appear green in the new color space. As a result, other scene elements reflected unnatural appearances. Coding was created in Java and X3D in XML form to accomplish these goals. A single matrix transformation-conversion to PC space-was used based on a single scene. The application of this transform to multiple scenes produced relatively consistent results. The initial goal of creating a high information density, ergonomic display of spectral imagery was partly met.

B. RECOMMENDATION FOR FUTURE WORK

Future work is needed to complete the design of a global eigenvector, which produces the transformation, and perhaps a non-linear hue mapping to facilitate more natural displays. Developing a non-linear hue mapping can be accomplished by utilizing a dynamic 3D scene instead of a current static 3D scene. X3D provides such options and needs to be developed further to enhance a proposed display strategy for Hyperspectral Images.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. ACRONYMS

ALRSS	Advanced Land Remote Sensing System
API	Application Program Interface
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
AWT	Abstract Window Toolkit
BIL	Band Interleaved by Line
BIP	Band Interleaved by Pixel
BSQ	Band Sequential
CCD	Charge Coupled Device
DN	Digital Number
DOM	Document Object Model
DTD	Document Type Definition
ENVI	Environment for Visualizing Images
HSI	Hyperspectral Imagery
HSV	Hue Saturation Value
HYDICE	Hyperspectral Digital Imagery Collection Experiment
JAMA	Java Matrix Package
JAXP	Java API for XML Parsing
LSF	Least Significant bit First
MSF	Most Significant bit First
MSS	MultiSpectral Spectrometer
NVIS	Night Vision Imaging System
PCT	Principal Component Transform
ROI	Region of Interest
TM	Thematic Mapper
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
X3D	Extensible 3D Graphics

XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language for Transformations

APPENDIX B. CONTENT OF CD ROM

1. DefaultHSVScatterPlot.x3d

This is the base X3D scene file. It should be placed where the Java application is invoked.

2. Data Flow chart.pdf.

This pdf file describes complete Remote Sensing process from end to end with data parameters.

3. Source code for Java application.

This Zip file contains the entire Java source code created for this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Apache XML Projects, [http://xml.apache.org/xalan-j/], October 2002.

Natural Resources Canada, Remote Sensing Tutorial, [http://www.ccrs.nrcan.gc.ca/ccrs/learn/tutorials/fundam/chapter1/chapter1_1_e.html], November 2002.

Ames, Andrea L., Nadeau, David R. and Moreland, John L., *VRML 2.0 Sourcebook*, New York: John Wiley & Sons, December 1997.

Anderson, R., Malila, Maxwell, W. and Reed, L., "Potential Defense Application of Multispectral and Hyperspectral Techniques", *Military Utility of Multispectral and Hyperspectral Sensors*, Infrared Information Analysis Center, Michigan, p. 7-1, November 1994.

Belokon, F. W. and et al., *Multispectral Imagery Reference Guide*, Fairfax, Virginia, Logicon Geodynamics, Inc., November 1997.

Brutzman, D. P., The Virtual Reality Modeling Language and Java, "*Communication of the ACM*", 41:6, pp. 57-64, March 1998.

Bunks, Carey, *Grokking the GIMP*, 2000, New Riders Publishing, [http://www.newriders.com], "Relating HSV to RGB", [http://gimp-savvy.com/BOOK/index.html?node52.html], November 2002.

Campbell, B. J., *Introduction to Remote Sensing*, New York, New York: Guillford Press, March 2002.

Deitel, H. M., Deitel, P. J., Nieto, T. R. and Lin, T. M., *XML How to Program*, Upper Saddle River, New Jersey: Prentice Hall, December 2001.

Diersen, D. I., *Colormetric Representations for Hyperspectral Imagery*, Master's Thesis, Naval Postgraduate School, Monterey, California, 2000.

Extensible 3D Task Group, [http://www.web3d.org/x3d.html].

Fortner, Brand, *The Data Handbook*, Second Edition, A Guide to Understanding the Organization and Visualization of Technical Data, Spinger-Verlag Publisher, March 1995.

Image Processing Toolbox User's Guide, Math Works, Inc., Natick, Massachusetts, May 1997.

Jackson, Richard, McDonald, Lindsay and Freeman, Ken, *Computer Generated Color*, John Wiley and Sons, New York, New York, January 1994.

JAMA, A Java Matrix Package, [http://math.nist.gov/javanumerics/jama/], September 2002.

Knudsen, Jonathan, *Java2D Graphics*, O'Reilly and Associates, Sebastopol, California, [http://www.oreilly.com/catalog/java2d/], May 1999.

Krauskopf, J., Williams, D. R. and Heeley, D. W., "Cardinal Directions of Color Space," Vision Resolution, 22, pp. 1123-1131, March 1982.

Liang, Y. Daniel, *Introduction to Java Programming with Jbuilder4*, Prentice Hall Inc., New Jersey, July 2002.

Lyon, Douglas A., *Image Processing in JAVA*, Prentice-Hall, Inc., p. 41, February 1999.

Moik, H., *Digital Processing of Remotely Sensed Images*: NASA no. 431, Washington, D.C., May 1980.

Parallelgraphics, Cortona Browser, [http://www.parallelgraphics.com/products/cortona/], October 2002.

Richards, John A., *Remote Sensing Digital Image Analysis*, Springer-Verlag Berlin Heidelberg, Germany, March 1993.

Rodrigues, Lawrence H., *Building Imaging Applications with Java Technology*, Addison-Wesley, May 2001.

Sanches, Julio and Canton, Maria P., *Space Image Processing*, CRC Press, p. 69, June, 1999.

Sabins, F. F., *Remote Sensing-Principals and Interpretation*, W. H. Freeman and Company, New York, March 1987.

Stelting, Stephen and Maassen, Olav, *Applied Java Patterns*, Sun Microsystems Press, Palo Alto, California, June 2002.

Swain, P. H., and Davis, S. M., 1978, *Remote Sensing-A Quantitative Approach*: McGraw-Hill Book Co., New York, March 2002.

Tyo, J. S., Dierson, D. I. and Olsen, R. C., "Development of an Invariant Display Strategy for Spectral Imagery," in Proceedings of SPIE vol. 4480: *Imaging Spectrometry* VI, M .R. Descour and S. S. Shen, eds., SPIE, Bellingham, Washington, August 2001.

Vane, G and Goetz, A. F. H., "Terrestrial Imaging Spectroscopy," *Remote Sensing of the Environment*, vol. 24, October 1988.

Xj3D Project, [http://www.xj3d.org], September 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Richard. C. Olsen, Code PH
   Department of Physics
   Naval Postgraduate School
   Monterey, California

4. Kang Suk Kim
   Paju-sy, Chori-myoun
   Osan-ry 225-5
   Kyongki-do
   Republic of Korea

5. Dr. Donald P. Brutzman, Code UW/BR
   Undersea Warfare Academic Group
   Naval Postgraduate School
   Monterey, California

6. William B. Maier II, Code PH
   Chair, Department of Physics
   Naval Postgraduate School
   Monterey, California

7. Peter J. Denning, Code CS
   Chair, Department of Computer Science
   Naval Postgraduate School
   Monterey, California

8. Moon, Young Han
   Major General, Defense Attaché
   Embassy of the Republic of Korea
   Washington, D.C.

9. Dr. Glen Wheless and Dr. Cathy Lascara
   VRCO, Inc.
   Virginia Beach, Virginia

10. Margaret Bailey
Applications Software Development
Sonalysts Inc.
Waterford, Connecticut

11. Cristina Russo dos Santos
ISITV -- Option Telecom
83162 LA VALETTE CEDEX

12. Dr. Taegong Lee
Korean National Defense University
Computer Science Department
Seoul, Republic of Korea

13. Dr Kim, Choelhwan
Korean National Defense University
Combat System Program
Seoul, Republic of Korea

14. Rudolf Panholzer
Chair, Space Systems Academic Group, Code SP
Monterey, California

15. Peter Chu
Professor, Code OC/CU
Monterey, California